Rahul Kaushal

kaushalrahul15@gmail.com · rahul-kaushal04 · rahulkaushal04 · rahulkaushal04.github.io/portfolio

Technical Skills

Languages: Python, SQL, Bash, Go
Cloud: Azure Cloud, Databricks
Frameworks: LLM/RAG, LangChain, FastAPI, PyTorch, Pydantic, Scikit-learn, PySpark, SQLAlchemy, Celery
Tools: Docker, Kubernetes, MLflow, PostgreSQL, Redis, Kafka, Git, GitHub Actions, Terraform, Prometheus

Education

Indian Institute of Technology Madras *July 2018 – June 2023*
B.Tech in Biotechnology & M.Tech in Data Science *CGPA: 8.77*

CBSE Std. XII *May 2016 – May 2018*
86.2%

Professional Experience

Software Engineer II *Jul 2024 – Present*
Honeywell — MLOps Platform Engineer

- Architected ForgeML, Honeywell's enterprise MLOps platform, as 8 production microservices with 30+ FastAPI REST endpoints covering the full ML lifecycle — training orchestration, model registration, security scanning, multi-target deployment, and real-time monitoring for multi-tenant workloads on Azure and Databricks.
- Engineered a multi-target deployment orchestration service for Databricks Serving Endpoints, AKS, and Edge/IoT devices with Git-based artifact packaging, rollback support, and dual async status-polling loops, reducing model deployment time from 3 days to 30 minutes for 15+ data scientists.
- Designed an AI-powered rule translation pipeline on Google Vertex AI (Gemini) with automated fine-tuning job submission, GCS training data upload, and async status polling, achieving 80%+ accuracy converting SkySpark Axon DSL rules into Honeywell Forge DSL.
- Built an asynchronous Azure Event Hub batch processing pipeline with Factory Pattern event dispatching, fault-isolated per-event parsing, and concurrent dual writes to PostgreSQL and Databricks Unity Catalog via `asyncio.gather`, cutting write latency by 50% with exactly-once semantics via Azure Blob checkpointing.
- Integrated 5 security scanners (Coverity, BlackDuckHub, TwistLock, ModelScan, AST Parser) using the Strategy Pattern alongside JWT RS256 authentication, OPA policy enforcement with 20+ fine-grained permissions, and a 7-layer SQL injection defense, catching 40+ vulnerabilities pre-deployment.
- Built a BDD automated acceptance testing framework (Behave/Gherkin) with parallel multi-tenant test execution via Python threading and Locust performance testing, covering onboarding, training, deployment, and scanning scenarios end-to-end.

Tools: *Python, FastAPI, Databricks, Azure Kubernetes (AKS), Google Vertex AI, MLflow, Azure Event Hub, PostgreSQL, PySpark, Docker, Prometheus*

Data Engineer II *Jun 2022 – Dec 2023*
Honeywell — Data Platform Engineer

- Built 15+ PySpark ingestion pipelines on Databricks pulling from SQL Server, Salesforce, SharePoint, and REST APIs into Delta Lake, using incremental watermark-based loading to avoid full re-scans on every run.
- Engineered a Spark Structured Streaming pipeline that ingested PDF files from Azure Blob Storage in real time, applied SCD Type 2 upsert logic to track document changes across micro-batches, and detected deleted files by cross-referencing Blob Storage with processed records.
- Built a sync layer that pushed curated Delta Lake data into PostgreSQL using batch upserts (1000 rows/batch) with `ON CONFLICT DO UPDATE` semantics, keeping the serving database in sync without full table reloads.
- Wrote SQL transformation pipelines for field service analytics, producing weekly/monthly/quarterly aggregations for incident resolution trends, contract health, maintenance metrics, and NPS scores consumed by the customer-facing DSP dashboard.
- Wired the data pipeline output into the RAG indexing layer by enabling Change Data Feed on Delta tables and triggering downstream Databricks vectorization jobs on incremental inserts, updates, and deletes.
- Contributed to the Maintenance Assist LLM chatbot by building the document vectorization pipeline into PGVector and an automated eval framework using Azure OpenAI as an LLM judge alongside BLEU/ROUGE scoring to measure chatbot response quality.

Tools: *PySpark, Databricks, Delta Lake, Apache Spark, PostgreSQL, Python, SQL, LangChain, PGVector, Azure OpenAI, Azure ML, Salesforce API, LangSmith, MLflow*

Projects

DepKeeper - Python Dependency Manager

2025 – Present

▷ Live Demo | ▷ GitHub

- Built a CLI tool to manage `requirements.txt` dependencies, supporting concurrent PyPI version checks via `asyncio`, with table, simple, and JSON output formats for CI/CD integration.
- Implemented a centralized PyPI metadata cache with double-checked locking to ensure each package is fetched at most once per run, shared across version checking and conflict resolution components.
- Designed a safe upgrade algorithm that strictly respects major version boundaries (e.g., never upgrades `2.x` to `3.x`), filtering candidates by Python compatibility before recommending updates.
- Engineered an iterative conflict resolver that detects cross-package version incompatibilities and attempts resolution within major version bounds, reverting to installed versions when no safe upgrade exists.

Tools: *Python, asyncio, httpx, Click, Rich, packaging (PEP 440/508), PyPI JSON API, pytest, GitHub Actions, MkDocs*

Hallownest Codex - Hollow Knight Save Analyzer

February 2026 – Present

▷ Live Demo | ▷ GitHub

- Built a full-stack dashboard that decrypts and visualizes Hollow Knight save files, tracking 200+ game state variables across charms, bosses, collectibles, and quests through an interactive web UI.
- Designed a layered data pipeline that safely transforms raw encrypted bytes into typed Python dataclasses, isolating all key parsing in one module to gracefully handle missing or version-mismatched save data.
- Developed a charm build optimizer using a knapsack-style solver with synergy scoring, producing goal-specific recommendations (damage, survivability, speedrun) constrained to the player's actual inventory and notch count.
- Integrated an interactive map with a custom tile system, slicing an 800MB source image into a zoomable pyramid with toggleable layers that reflect live save state for 100+ in-game collectibles and bosses.
- Shipped a spoiler-guard system as a global toggle that conditionally hides item locations, hints, and quest steps behind blur overlays, with per-component wrappers for fine-grained content control.

Tools: *Python, Streamlit, Folium, Plotly, Pillow, pandas, PyCryptodome, ReportLab*