Haikal Hilmi - Data Engineer

Data Engineer specializing in large-scale scraping, resilient data pipelines, ETL, search infrastructure, observability, and on-premise HPC. Experienced processing high-volume social, news, video, and business data in production.

This document is optimized for search, retrieval-augmented generation (RAG), and AI knowledge bases. It is generated from the live portfolio data source.

Contact and profiles

- Website: https://haikalhilmi.my.id
- LinkedIn: https://www.linkedin.com/in/haikalhilmi/
- Upwork: https://www.upwork.com/freelancers/~01ff216c8a0f8a68a1
- Project inquiry: https://tally.so/r/81VNPz

Core capabilities

- Data pipelines: ingestion, ETL, orchestration, queues, search, monitoring, and failure recovery.
- Web scraping: high-volume collection, anti-fragile workers, scheduling, enrichment, and structured delivery.
- Infrastructure: on-premise HPC, Docker workloads, Elasticsearch, PostgreSQL, Grafana, and Prometheus.
- Applied AI: sentiment analysis, speech-to-text, entity extraction, retrieval, and data enrichment.

Selected scale and infrastructure

- 8 HPC Servers - 4 main + 4 additional
- 64 Cores (max) - AMD EPYC 7502P
- 200GB+ RAM (max) - Ubuntu 22.04 / CentOS
- 100K+ Records / day - Pipeline throughput
- 4x Tesla V100 GPUs - NVIDIA DGX Station
- 128GB VRAM (total) - 32GB per GPU

Professional experience

ByteDance

- Role: Backend Engineer Intern
- Period: Feb 2025 - Jun 2025
- Location: Yogyakarta, ID
- Impact: Built a microservice handling 100,000+ requests/second.
- Context: Contributed to Tokopedia x TikTok-Shop system integrations.
- Technologies: Go, gRPC, RocketMQ, NSQ, Bytecloud
- Highlights:
- Contributed to Tokopedia x TikTok-Shop system integrations.
- Added Lark support to the internal dev kit, cutting log spam ~60%.

Semesta Data Digital

- Role: Data Engineer Contract
- Period: Jan 2024 - Nov 2025

- Location: Jakarta, ID
- Impact: Designed HPC data pipelines - reliability +25%, processing time 40%.
- Context: Operated an 8-server HPC cluster ingesting social-media & news data at scale.
- Technologies: Python, Elasticsearch, PostgreSQL, RabbitMQ, Docker, Grafana
- Highlights:
- Architected a resilient social-media data pipeline ingesting from diverse sources.
- Built Grafana + Prometheus monitoring with alerts that prevented server crashes.

Tilikan Indonesia

- Role: Web App Developer Contract
- Period: Sep 2023 - Mar 2024
- Location: Yogyakarta, ID
- Impact: Analytics dashboards - reliability +50%, cross-team efficiency +70%.
- Technologies: Next.js 14, React, TypeScript, MongoDB
- Highlights:
- Integrated multiple APIs to streamline interdepartmental workflows.
- Shipped a document-conversion feature for better document management.

Data engineering projects

Android Ad Evidence Capture & DSP Attribution

- ID: android-ad-evidence
- Category: Data Engineering
- Context: client
- Role: Automation Engineer / Python Engineer
- Status: Android advertising verification PoC
- Scale: 463 flows 10 ad networks 7-minute lab session
- Summary: An automated Android ad-inspection system that captures creatives, UI evidence, click destinations, and network traffic to verify delivery paths and investigate DSP attribution such as Smadex.
- Problem: A screenshot cannot prove which buying platform delivered an ad, while dynamic creatives, stale intents, certificate restrictions, freezes, and non-rooted devices make reliable evidence collection difficult.
- Solution: Built a dual-path pipeline: ADB and uiautomator2 record what users see and safely inspect interactions, while mitmproxy captures delivery-chain signals for rule-based correlation and tamper-evident archiving.
- Result: Tamper-evident evidence bundles 95.69% coverage across 90 tests.
- Technologies: Python 3.11, uiautomator2, ADB, mitmproxy, Pillow, ImageHash, PowerShell, Pytest, mypy, JSON Lines, SHA-256
- Source & evidence: https://github.com/Harmerz/Android-advertising-verification-PoC
- Pipeline: Launch: PowerShell + Python CLI -> Control & detect: ADB + uiautomator2 -> Capture evidence: UI + screenshots + safe clicks -> Capture network: ADB reverse + mitmproxy -> Attribute & archive: Rules + JSONL + SHA-256
- Key decisions:
- Kept UI and network capture independent: Android automation proves what the user saw, while intercepted flows provide delivery-chain evidence.
- Required explicit Ad Info, redirect, client-signature, or reporting signals for confirmed DSP attribution; creatives alone remain candidates, never proof.
- Captured broadly, matched later, and recorded creative URLs without fetching them-preserving limited-window evidence without creating accidental impressions or clicks.
- Externalized provider indicators in JSON and separated attribution into confirmed, candidate, and notdetected states.

- Protected artifacts with append-only SHA-256 manifests, recorded measured geo metadata, and added ANR recovery, cold restarts, and proxy restoration for long sessions.
- Portfolio detail: https://haikalhilmi.my.id/projects/android-ad-evidence

Competitor SEO Monitoring Pipeline

- ID: competitor-seo-monitoring
- Category: Data Engineering
- Context: owned
- Role: Data / Backend Engineer
- Status: Portfolio project Local-first PoC
- Scale: 40 live fetches 25 complete articles 0 failures
- Summary: A local-first pipeline for monitoring competitor content and SEO changes, using Daypass as the primary website and ResortPass as the competitor.
- Problem: Recurring competitor SEO reviews are slow and hard to audit when assembled manually, while raw HTML comparisons create noisy alerts from navigation, timestamps, and dynamic widgets.
- Solution: Built two isolated paths: a free crawler, snapshot, diff, dashboard, and workbook workflow; plus optional DataForSEO enrichment guarded by dry runs, environment-only credentials, caching, and a hard cost ceiling.
- Result: Complete article records paid enrichment capped below $1.
- Technologies: Python 3.11+, JSONL / JSON, DataForSEO API v3, Node.js, @oai/artifact-tool, HTML, CSS, JavaScript, unittest
- Source & evidence: https://github.com/Harmerz/Competitor-SEO-Monitoring-Pipeline
- Pipeline: Discover: robots.txt + sitemaps + links -> Crawl: Bounded Python crawler -> Extract: Metadata + normalized content -> Store & compare: Immutable JSONL + diffs -> Enrich: Approved DataForSEO requests -> Report: Dashboard + auditable XLSX
- Key decisions:
- Used standard-library Python for the core crawler so the collection path stays reproducible without third-party Python dependencies.
- Chose robots-aware sequential crawling for ethical, auditable rate control instead of maximum throughput.
- Compared normalized main text for meaningful changes; raw HTML hashes remain diagnostic signals only.
- Made dated JSONL snapshots immutable unless replacement is explicitly requested with --force.
- Put DataForSEO behind a trust boundary: dry-run by default, environment-only credentials, allowlisted hosts, immutable caching, and a cost check before every request.
- Never fabricates missing SEO metrics; workbook formulas reference raw-data sheets and the dashboard binds only to loopback.
- Portfolio detail: https://haikalhilmi.my.id/projects/competitor-seo-monitoring

Cross-Platform Customer Review Scraper

- ID: cross-platform-review-scraper
- Category: Data Engineering
- Context: owned
- Role: Data / Backend Engineer
- Status: Portfolio project Cross-platform review extraction
- Scale: 3 review platforms 25 sample records
- Summary: A Python scraping toolkit that collects public customer reviews and app metadata from Trustpilot, Google Play, and the Apple App Store for customer research and sentiment analysis.
- Problem: Review research is fragmented across platforms with different delivery models: dynamic Next.js data on Trustpilot, paginated tokens on Google Play, and limited page-rendered public review data on the App Store.
- Solution: Built isolated scrapers for each source, then normalized review text, ratings, authors, dates, metadata, and source URLs into analysis-friendly JSON with deterministic deduplication and committed samples.
- Result: Normalized review JSON source URLs preserved.

- Technologies: Python 3.11+, Patchright, google-play-scraper, Apple iTunes Lookup API, App Store public web JSON, JSON, PowerShell, Google-style docstrings
- Source & evidence: https://github.com/Harmerz/Cross-Platform-Customer-Review-Scraper
- Pipeline: Discover: Profile URLs + package/app IDs -> Fetch: Patchright + platform endpoints -> Extract: Reviews + app/business metadata -> Normalize: Analysis-friendly JSON records -> Deduplicate: Review IDs + fallback keys -> Store: JSON outputs + trimmed samples
- Key decisions:
- Kept one script per platform so source-specific fetching, pagination, and fallback behavior remain isolated and maintainable.
- Used a Patchright browser context for Trustpilot before reading public Next.js JSON, reusing a realistic session without scraping visual markup.
- Prioritized App Store page-rendered JSON and retained the legacy RSS path only as a fallback because the public RSS feed is often empty.
- Preserved source URLs, deterministic review IDs, and fallback text/date keys so records remain auditable and deduplicated across runs.
- Stored trimmed portfolio samples separately from full generated outputs and documented platform limitations rather than hiding or estimating missing coverage.
- Portfolio detail: https://haikalhilmi.my.id/projects/cross-platform-review-scraper

Social Media Data Pipeline

- ID: pipeline
- Category: Data Engineering
- Context: client
- Role: Data Engineer
- Scale: Hundreds of thousands of records / day
- Summary: The pipeline behind the social-media & news scrapers - powering two-sided sentiment analysis and NER for a public-sector communications team.
- Problem: Data volume is enormous and must be processed continuously.
- Solution: HPC pipeline with monitoring and a queue system.
- Result: Cut processing time ~40% and improved reliability ~25%, monitored 24/7.
- Technologies: RabbitMQ, Docker, Elasticsearch, Grafana, Prometheus, Airflow
- Pipeline: Sources: Scrapers -> Queue: RabbitMQ -> Process: Docker workers -> Store: Elasticsearch -> Monitor: Grafana + Prometheus
- Key decisions:
- RabbitMQ for flexible routing - speed from scrape to display wasn't the bottleneck, so simple routing beat Kafka's complexity.
- Elasticsearch for fast full-text & context search across large volumes of news and social text.
- An alert bot pings the dev team the moment a scraper returns no data - kept uptime ~99% with fast recovery.
- On-prem HPC: the ~40% efficiency gain freed capacity for co-located AI and queue workloads.
- Portfolio detail: https://haikalhilmi.my.id/projects/pipeline

Instagram Scraping

- ID: instagram
- Category: Data Engineering
- Context: client
- Role: Data Engineer (solo)
- Scale: Up to ~50K posts & ~1M comments / day
- Summary: Large-scale Instagram data collection for AI sentiment analysis and competitor monitoring.
- Problem: Hard to obtain competitor engagement data automatically.
- Solution: Mass scraping of Instagram posts & comments.
- Result: Clean, structured engagement data-delivered daily.

- Technologies: Python, Selenium, PostgreSQL, Airflow
- Pipeline: Source: Instagram -> Crawl: Selenium -> Orchestrate: Airflow -> Store: PostgreSQL -> Analyze: Sentiment AI
- Key decisions:
- Benchmarked CPU, RAM, disk, and network per scraper - part of my undergraduate thesis research.
- Portfolio detail: https://haikalhilmi.my.id/projects/instagram

Twitter / X Scraping

- ID: twitter
- Category: Data Engineering
- Context: client
- Role: Data Engineer (solo)
- Scale: Up to ~75K tweets / day
- Summary: Distributed Twitter/X scraping for AI sentiment analysis and public-conversation monitoring.
- Problem: Client needed real-time monitoring of public conversations.
- Solution: A distributed Twitter crawler with a queue system.
- Result: A real-time pulse on public conversation.
- Technologies: Python, Selenium, RabbitMQ, Elasticsearch
- Pipeline: Source: Twitter / X -> Crawl: Selenium -> Queue: RabbitMQ -> Store: Elasticsearch -> Analyze: Sentiment AI
- Portfolio detail: https://haikalhilmi.my.id/projects/twitter

YouTube Scraping

- ID: youtube
- Category: Data Engineering
- Context: client
- Role: Data Engineer
- Scale: Up to ~5K videos & ~500K comments / day
- Summary: YouTube video and comment intelligence with automatic transcription.
- Problem: Needed large-scale analysis of videos and comments.
- Solution: Video & comment crawling pipeline + automated transcript analysis.
- Result: Searchable video insights, transcripts included.
- Technologies: Python, Elasticsearch, RabbitMQ, YouTube API
- Pipeline: Source: YouTube API -> Crawl: Python -> Queue: RabbitMQ -> Transcribe: Speech-to-Text -> Store: Elasticsearch
- Portfolio detail: https://haikalhilmi.my.id/projects/youtube

News Scraping

- ID: news
- Category: Data Engineering
- Context: client
- Role: Data Engineer
- Scale: 226K+ articles / week
- Summary: A unified crawler across ~2,900 news portals - automatic extraction for most via newspaper4k, with a custom-parser fallback for ~100 sources it couldn't handle out of the box.
- Problem: newspaper4k covers most portals with just a URL, but ~100 sources have markup it can't parse correctly.
- Solution: Built a reusable custom-parser template, adapted per portal with small tweaks instead of one-off scripts for each.
- Result: One consistent, clean feed across ~2,900 portals, with a maintainable fallback layer for the long tail.
- Technologies: Python, BeautifulSoup, Selenium, newspaper4k

- Pipeline: Sources: News portals -> Crawl: BeautifulSoup / newspaper4k -> Extract: Content parser -> Store: Database -> Analyze: NER + Sentiment AI
- Portfolio detail: https://haikalhilmi.my.id/projects/news

Lead Generation

- ID: leadgen
- Category: Data Engineering
- Context: client
- Role: Data Engineer
- Scale: Thousands of leads
- Summary: Automated B2B and B2C prospecting-email and company data on demand.
- Problem: Finding business prospects manually is hard.
- Solution: Automated email & company-data discovery.
- Result: Thousands of qualified prospects, sourced automatically.
- Technologies: Python, Automation, Apollo, ZoomInfo
- Pipeline: Sources: Apollo / ZoomInfo -> Automate: Python -> Enrich: Dedup + verify -> Deliver: CSV / CRM
- Portfolio detail: https://haikalhilmi.my.id/projects/leadgen

In-Video Word Finder

- ID: kick
- Category: Data Engineering
- Context: client
- Role: Sole Engineer
- Scale: 100+ hours of video processed
- Summary: Finds the exact moment a streamer says a target word (e.g. "LFG") for marketing clips - speech to searchable text via AI.
- Problem: A marketing client needed to catch specific words spoken across long Kick.com streams.
- Solution: Transcribe with Whisper, index words with timestamps, then auto-cut the clip at that moment.
- Result: Delivers the exact clip + URL where each searched word is spoken.
- Technologies: Python, Whisper, GPU
- Pipeline: Source: Kick.com -> Extract: Video -> Audio -> Transcribe: Speech-to-Text AI -> Index: Searchable text -> Output: Word finder
- Key decisions:
- Ran Whisper (medium) locally on a laptop GPU - accurate enough for keyword search at zero STT API cost.
- Batch-processed 100+ hours of video; indexed transcripts with timestamps to auto-cut clips.
- Whisper's language detection handled mixed-language streams.
- Portfolio detail: https://haikalhilmi.my.id/projects/kick

Auto-Engagement Bot

- ID: twitter-bot
- Category: Data Engineering
- Context: client
- Role: Data Engineer
- Scale: Hundreds of comments / day
- Summary: Automated, on-topic commenting built on the Twitter/X scraping system.
- Solution: Scheduled automated commenting on relevant conversations.
- Result: Hands-free engagement at scale.
- Technologies: Python, Automation
- Pipeline: Source: Twitter / X -> Filter: Relevance match -> Schedule: Python -> Act: Auto-comment
- Portfolio detail: https://haikalhilmi.my.id/projects/twitter-bot

Client testimonials

"As commissioner of Semesta Data Digital and Global Data Inspirasi, I've seen first-hand how Haikal designs data infrastructure, runs large-scale pipelines and scraping with high reliability, and helped shape products like Robota. He was also my teaching assistant for a year - consistent, a fast learner, and highly dependable. He combines genuine technical depth with a strong sense of ownership."

- Widyawan, S.T., M.Sc., Ph.D., Commissioner, Semesta Data Digital & Global Data Inspirasi; Yogyakarta, ID (profile (https://www.linkedin.com/in/widyawan/))

"I had the pleasure of working with Haikal on a scraping project, and he exceeded my expectations in every aspect. Professional, responsive, and highly skilled, delivering high-quality work on time. I highly recommend Haikal - he's a fantastic contractor."

- Jeesun Kim, Upwork client - Twitter/X Scraper; Aurora, Canada

"Excellent service, quick responsiveness, and great technical skills. Haikal developed an effective and high-performing URL scraper, offered a 3-month warranty, and I would highly recommend him to others."

- Sam White, Upwork client - URL Scraper - 5.0 review; United States

"Haikal is very smart, creative and good at what he does. He is a great communicator and very efficient."

- Claire Bartolozzi, Upwork client - Tweet Collector; South Yarra, Australia

"Haikal was very quick to respond and professional in his work. He even suggested some improvements to the deliverables, which I found useful. I would definitely work with him again."

- Cristian Buda, Upwork client - PDF to JSON Extraction; Cluj-Napoca, Romania

Source and freshness

- Canonical portfolio: https://haikalhilmi.my.id/data-engineer
- Portfolio scope: de
- Markdown export: https://haikalhilmi.my.id/portfolio.md?scope=de
- PDF export: https://haikalhilmi.my.id/portfolio.pdf?scope=de
- AI discovery file: https://haikalhilmi.my.id/llms.txt
- The export is generated from the same typed project and experience data used by the website.