

# Table of Contents

<b>1</b>	<b>Technical Competencies</b>	<b>2</b>
1.1	Current Role & Production Systems	2
1.2	Career Background & Experience	2
1.3	Key Production System	2
1.4	Relevant Technical Expertise	2
1.4.1	Python Automation & Workflow Systems	2
1.4.2	Cloud & Container Infrastructure	3
1.4.3	System Programming & Performance	3
1.5	Production Examples Relevant to Netstrata	3
1.5.1	1. Migration Infrastructure Expertise	3
1.5.2	2. Data Quality & Integrity Systems	3
1.5.3	3. Internal Automation Tools	4
1.5.4	4. AI-Augmented Development Tools (Production Context)	4
1.6	Technical Environment & Practices	4
1.6.1	Development Workflow	4
1.6.2	Code Quality Standards	5
1.6.3	Rust Production Experience	5
1.6.4	Rapid Prototyping Capability	5
1.7	What This Means for Netstrata	5
1.7.1	Phase 1 Contribution (Software Completion)	5
1.7.2	Phase 2 Contribution (WA Migration Readiness)	6
1.7.3	Phase 3 Contribution (External Rollout - if still contributing post-2026)	6
1.8	Technical Philosophy	6

# 1 Technical Competencies

## 1.1 Current Role & Production Systems

Director of Operations at Eon Labs Ltd. ([www.eonlabs.com](http://www.eonlabs.com)), handling administration, HR, SR&ED (Scientific Research and Experimental Development), feature engineering for quantitative trading systems, and AI agent coding tool adoption across the team. Focus areas include Claude Code Agent Skills popularization, automation workflows, and cloud infrastructure for financial technology systems.

Seeking final career opportunity in team environment at established business, transitioning from independent operations role to hands-on technical contribution.

## 1.2 Career Background & Experience

**25 years of SaaS development experience** including founding and scaling internet solutions company (2000) to 100+ employees across Hong Kong, Shanghai, and Macau. Built first-generation web analytics tools (pre-Google Analytics era). Understand both scaling challenges and market positioning importance from operator perspective.

Learned critical lessons about technology adoption patterns, market timing, and the importance of established markets with proven customer bases—which directly informs interest in contributing to Netstrata’s established position in NSW strata management with proprietary software completing 2026.

## 1.3 Key Production System

Operational automation system using Telegram bot interface for team workflow orchestration (quantitative trading infrastructure monitoring):

- **launchd supervision:** Automatic crash detection and restart
- **100ms auto-reload:** Rapid iteration and deployment
- **Doppler credential management:** Secure secrets handling
- **24/7 operation:** Continuous monitoring and maintenance

Demonstrates operational reliability patterns applicable to business-critical systems.

## 1.4 Relevant Technical Expertise

### 1.4.1 Python Automation & Workflow Systems

- **Package Management:** `uv` for modern Python dependency management (avoiding legacy pip/conda)
- **Self-Contained Scripts:** PEP 723 inline dependencies for portable automation tools

- **Production Deployment:** launchd/systemd service management for unattended operation
- **API Integration:** Claude API, web scraping (Playwright + BeautifulSoup), third-party service orchestration

**Recent Example:** Extracted 292 blog posts from Netstrata website using automated web scraping with AJAX pagination handling, rate limiting, and metadata preservation.

#### 1.4.2 Cloud & Container Infrastructure

- **Container Runtime:** Colima + Docker CLI on macOS for local development and testing
- **Cloud Platforms:** Experience with cloud-native deployment patterns
- **Infrastructure as Code:** Configuration management for reproducible deployments

#### 1.4.3 System Programming & Performance

- **Rust:** Modern systems programming with cargo toolchain
- **Testing:** cargo nextest for fast test execution
- **Pre-commit Hooks:** Automated code quality checks before deployment

### 1.5 Production Examples Relevant to Netstrata

#### 1.5.1 1. Migration Infrastructure Expertise

Expertise in data transformation patterns, validation systems, and rollback mechanisms for enterprise migrations:

- **Data Transformation:** Converting legacy formats to modern schemas with integrity preservation
- **Validation Frameworks:** Automated testing of data completeness and consistency
- **Rollback Mechanisms:** Safe deployment strategies with recovery procedures
- **Documentation Automation:** Systematic migration guide generation

**Relevance to Phase 2:** WA customer migration requires similar infrastructure (migration scripts, validation, support documentation).

#### 1.5.2 2. Data Quality & Integrity Systems

Published open-source package (PyPI: gapless-crypto-data v3.3.0) handling large-scale data collection with quality guarantees:

- **Gap Detection:** Automated identification of missing data in time-series datasets
- **Atomic File Operations:** Corruption-resistant writes using temporary files and atomic renames
- **Data Validation:** Integrity checks ensuring completeness and consistency
- **Corruption Prevention:** Atomic file operations preventing partial writes

**Published Package:** MIT-licensed on PyPI, 22x faster than REST API polling through intelligent failover to Binance public repository with data validation pipelines.

**Relevance to Phase 2:** WA migration requires data integrity assurance across scheme records, financial history, and compliance documents. Gap detection and atomic operations prevent data loss during multi-stage migration.

### 1.5.3 3. Internal Automation Tools

Created workflow automation systems that:

- **Eliminate Manual Data Entry:** Bulk operations replacing repetitive form filling
- **API Orchestration:** Coordinating multiple third-party services
- **Report Generation:** Automated document creation from structured data
- **Notification Systems:** Real-time alerts via Telegram/email for operational events

**Relevance to Phase 1:** Andrew Tunks' operational efficiency mandate aligns with automation capabilities.

### 1.5.4 4. AI-Augmented Development Tools (Production Context)

Extensive production experience with AI coding agents:

- **Claude Code CLI:** 1000+ hours with Agent Skills development, custom workflow design
- **GitHub Copilot:** AI-assisted code completion and suggestion patterns
- **Codex:** OpenAI's code generation model for AI-powered development
- **Cursor:** AI-augmented IDE with context-aware code generation
- **Claude API:** Large-scale document processing (200,000 tokens), tool use integration, production automation

**Focus areas:** Prompt engineering, context window management, systematic workflow integration, reliability patterns, cost management

**NOT speculative AI hype** - this is production experience with measurable outcomes (hours saved, errors reduced, processing speed). Detailed AI methodology described in attached document (01b).

## 1.6 Technical Environment & Practices

### 1.6.1 Development Workflow

- **Unix-Like Systems:** macOS/Linux (POSIX shells, standard conventions)
- **Version Control:** Git with pre-commit hooks for code quality
- **Testing:** Automated test suites with fast feedback loops
- **Documentation:** Machine-readable specs (OpenAPI, JSON Schema, YAML)

### 1.6.2 Code Quality Standards

- **Dependency Auditing:** cargo deny check for security vulnerabilities
- **Automated Testing:** Continuous validation before deployment
- **Error Handling:** Graceful failure and recovery patterns
- **Observability:** Logging, monitoring, crash detection

### 1.6.3 Rust Production Experience

**rangebar** - Published on Crates.io (v2.0.0, 1,863 downloads) - financial data processing library:

- **Workspace Architecture:** Modular crate design with 8 sub-crates for maintainability
- **Algorithm Specification:** Formal mathematical specifications with invariant testing
- **Documentation Management:** Authoritative spec pattern (single source of truth for algorithm behavior)
- **Production Patterns:** Fixed-point arithmetic, non-lookahead bias guarantees, temporal integrity

#### Documentation Standards:

- Centralized specifications replacing duplicated documentation
- Version-tracked algorithm specs with breaking change history
- Comprehensive testing validating conformance to specifications

**Relevance to Phase 1:** Software completion requires documentation discipline and specification rigor. Authoritative spec patterns prevent divergence between code, tests, and documentation across Tom Bakani's team.

### 1.6.4 Rapid Prototyping Capability

**Speed:** MVPs in days, not months **Iteration:** Fast feedback cycles with 100ms reload times **Production Quality:** From prototype to reliable operation quickly

**Example:** Netstrata blog extraction project went from concept to 292 extracted posts in under 48 hours.

## 1.7 What This Means for Netstrata

### 1.7.1 Phase 1 Contribution (Software Completion)

- **Modern Tooling:** Python/uv, Rust for performance-critical components
- **Automation Expertise:** Reducing manual operational overhead
- **Production Reliability:** Experience with 24/7 supervised systems
- **Fast Iteration:** Quick turnaround on features and bug fixes

### 1.7.2 Phase 2 Contribution (WA Migration Readiness)

- **Migration Infrastructure:** Tools for data conversion, validation, rollback
- **Support Systems:** Documentation generation, training materials
- **Testing Frameworks:** Automated validation of migration accuracy
- **Deployment Automation:** Reproducible rollout procedures

### 1.7.3 Phase 3 Contribution (External Rollout - if still contributing post-2026)

- **Packaging:** Deployment automation for external customers
- **Distribution Systems:** Update mechanisms, version management
- **Support Infrastructure:** Issue tracking, documentation, monitoring
- **Freemium Mechanics:** Usage limits, feature gating, upgrade paths

## 1.8 Technical Philosophy

### Focus on Business Value:

- Automation should save measurable time (hours → minutes)
- Systems should run unattended (launchd supervision, crash recovery)
- Tools should be maintainable (clear code, good documentation)
- Technology serves operations (not technology for its own sake)

### Avoid Hype:

- No claims about AI capabilities without production evidence
- Focus on solved problems, not potential future benefits
- Honest about limitations and trade-offs
- Measure outcomes (time saved, errors reduced, uptime)

### Join Existing Team:

- Work with Tom Bakani's software operations team
- Support Andrew Tunks' operational efficiency mandate
- Contribute to end-2026 completion milestone
- Team member, not external auditor