IAN CLARK

ian@ianleeclark.com

TECHNICAL SKILLS

Languages	Python, Go, Kotlin, Javascript, Elixir
Frameworks	Vue.js, Flask, Spring, Phoenix
Datastores	MySQL, Redis, Postgres, Elasticsearch
Tools & Miscellaneous	AWS, Docker, RabbitMQ, Celery, Kafka
Github	https://github.com/GrappigPanda

EXPERIENCE

LeMans Corp. (Industry: ECommerce) *First 3 months remote*

Software Engineer

Utilizing Elasticsearch to serve as the primary search engine within our e-commerce platform–storing and indexing over 20k products with 480k derivatives of those products.

Writing an Intelligence Service in Kotlin and Spring which aggregates data from disparate services and performs various statistical analysis to aide search relevancy and related products.

Writing ETL pipelines to retrieve and sync e-commerce data from legacy (MSSQL) databases into our modern platform databases (MariaDB, Postgres, and ElasticSearch).

Architecting and programming API gateways and microservices in Go and Kotlin.

Successfully made a push for more integration tests to ensure API compatibility, changes are non-breaking, and that customers will not be using a broken product.

VirtuCrypt (Industry: Cloud-Based Cryptography)

Web Application Developer (Full Stack) & Cryptographic Key Manager

Rewriting a TCP multiplexer in Go for a 3200% command throughput gain over Python.

Creating a new service offering (Transparent Encryption Layer) allowing for customers to encrypt and decrypt files on-demand utilizing third-party storage providers (AWS S3, Box, Google GCS, &c.) without giving access to encryption/decryption keys.

Developing a new database application layer allowing the back-end API to transition from using a third-party CRM (Sugar) to Postgres greatly improving back-end and front-end performance and saving the company over 10,000$ per year.

Architecting and building a customer-facing RESTful API (Python, Flask, Postgres) allowing cloud-based asymmetric key management, cryptography for point-of-sale devices, debit processing, and remote key injections.

Building customer-facing dashboards to utilize the back-end API in Vue.js.

PERSONAL PROJECTS

Notorious Primary language: Go

Utilized technologies: Redis, MySQL, Docker

A feature-complete Bittorrent tracker used to serve Linux and BSD OS distributions. Implemented in Go using Redis to store and quickly serve (short-term) peer information, MySQL or Postgres to track (long-term) peer information, and Docker for Continuous Integration and Continuous Deployment.

Olivia Primary language: Go

Utilized technologies: Bloom filters, Distributed Hash Tables

A Distributed, NoSQL key-value storage utilizing bloom filters for efficient key searches throughout the network and built upon a distributed hash table.

EDUCATION

The University of Texas at San Antonio
Bachelor of Business Administration in Economics Date of Graduation: May 2015

ENTREPRENEURIAL ENDEAVORS

ProvideBooking.com
Stack: Python 3.6, Flask, Vue.js, Postgres, Docker, AWS

Automating staging and production deployments to DigitalOcean through Docker Compose

Architecting and individually engineering an MVP capable of allowing companies/clients/users to pay and schedule their contractors/freelancers/users in one place.

Writing the front-end dashboard in Vue.js + VueX (Redux-like store)

Reaching out to over 200 potential customers in initial targeted niche to gather feedback and determine which features are most desired in an MVP.

Setting up Drone CI on DigitalOcean to ensure pull requests and commits aren't breaking any unit tests, end-to-end tests, or integration tests.

Genuinely proud of this code-base, so walk-through requests during interviews are highly welcome!

SMSQueue.io
Stack: Elixir + Phoenix, Caddy, Postgres, & Kotlin

Building a simple service capable of supplementing Provide Booking by automating the sending of SMS appointment reminders.

Architecting a serverless environment capable of handling SMS processing in an insanely scalable and cost-effective manner.

INDEPENDENT CONTRACTING

Postgres Optimizations

Automating nightly vacuum analyze to ensure proper postgres statistics were available for index scans resulting in around a 50% improvement for queries.

Implementing table partitioning (6 month chunks to ensure partitioned table sizes between 5 and 10 million rows) to drop query times from 160 seconds to 3 second.

Highest improvement in queries was a drop from 350 seconds to 3271ms.

Trello to Mechanical Turk Integration

Architecting a serverless (AWS Lambda, API Gateway, SQS, CloudWatch) platform capable of handling Trello webhooks, creating Mechanical Turk jobs, and reporting back status whenever those jobs were completed. Each of these steps being an individual Lambda function.

Utilizing this architecture enabled the client to not worry about deployment and is keeping overhead much cheaper than even the smallest of EC2 instances.