Mohamed Bana's Curriculum Vitae

Senior Software Engineer

Contact

+44-7960-045-281
mohamed@bana.io
https://stackoverflow.com/users/241993/mohamed-bana

Web

https://github.com/mbana
https://linkedin.com/in/mbana
https://bana.io/blog

I am a highly skilled Software Engineer with 13 years of job experience with a proven record. I love to build and maintain high-availability rock-solid systems that support successful businesses. I am looking for an interesting **remote-only** job, solo or in a small team of professionals to share my knowledge and to learn from. Part-time occupation is negotiable. Please read my cover letter at **https://bana.io/resume/cover-letter**. To download my CV and/or cover letter, please see **https://bana.io/resume/cv-download**.

Work Experience

Senior Software Engineer (Remote), Cynergy Bank, London, United Kingdom - 03/2023–07/2023

Tech: PostgreSQL, Postgres, Continuous Integration and Continuous Delivery (CI/CD), OpenAPI Specification (OAS), Swagger API, Google Cloud Platform (GCP), REST APIs, go, Linux, Golang, Go (Programming Language), Docker, Git, Google Cloud Platform and Kubernetes.

Senior Software Engineer (Remote), Kubeshop, Greenwich, United States - 05/2022–01/2023

Worked on kusk-gateway, a OpenAPI-driven API Gateway for Kubernetes. Links:

- **https://kusk.kubeshop.io/**
- **https://github.com/kubeshop/kusk-gateway**

Kusk is a Kubernetes API gateway powered by Envoy. The main difference with other API Gateways is that Kusk has native support for OpenAPI.

Modern REST APIs are developed using OpenAPI specification that is then used to generate API documentation, tests, server stubs and clients all from the OpenAPI definition. Kusk enables the use of OpenAPI definitions to configure the Ingress Controller of your Kubernetes clusters.

The commits I made against the repository is available to view at: **https://github.com/kubeshop/kusk-gateway/commits?author=mbana**.

Tech: Golang, Kubernetes, Kubernetes Control Plane, go-control-plane, Envoy Proxy, Docker, Docker Compose, Minikube, Shell scripting/Bash, gRPC, Protocol Buffers, GitHub Workflows, Linux.

Senior Software Engineer (Remote), ViTRiFi Limited, London, UK - 11/2021–03/2022

Cannot disclose details due to NDA.

Tech: Golang, Kubernetes, Docker, Docker Compose, AWS, Amazon EKS, Grafana, Loki, Prometheus, Shell scripting/Bash, Visual Studio Code Remote - Containers, gRPC, Protocol Buffers, Kafka, Redpanda Kafka, GitLab.

Software Engineer (Remote), Paymentsense Limited, London, UK - 07/2021–10/2021

I worked with Golang on Connect-E (**https://docs.connect.paymentsense.cloud/ConnectE/GettingStarted**).

Tech: Golang with modules, Docker, Docker Compose, TypeScript, GCP, Google Cloud Datastore, Google Cloud Big Query, Google Cloud Pub / Sub.

Software Engineer (Remote), IBM, Winchester, UK - 09/2020–04/2021

I worked on IBM Cloud as Software Engineer on the IKS Cluster (IBM Cloud™ Kubernetes Service) team as Cloud Software Engineer / Golang Engineer. IKS is effectively something like AWS, GCP or Azure, see IBM Cloud.

- I'm a member of the IBM Cloud™ Kubernetes Service team responsible for delivering IBM's Kubernetes managed container service. As a certified K8s provider, IBM Cloud Kubernetes Service provides intelligent scheduling, self-healing, horizontal scaling, service discovery and load balancing, automated rollouts and rollbacks, along with secret and configuration management and a fully managed image registry with integrated vulnerability scanning.
- Working in an agile way and operating with a continuous delivery model.
- Team/Squad consisted of around nine (9) people and we managed the complete life cycle of deliveries.

Tech: Go, Golang, Shell Scripting, Bash, Docker, Docker Compose, Kubernetes, RedHat, Travis CI, LogDNA, go.uber.org/zap, etcd or What is etcd? | IBM.

Full Stack Software Engineer, Open Banking Limited, London, UK - 05/2018–01/2020

Working as a full stack software engineer at Open Banking on a tool that will validate a bank's implementation of the OpenBanking API spec, see:

- **https://github.com/OpenBankingUK/conformance-suite**
- **https://bitbucket.org/openbankingteam/conformance-suite**
- **https://hub.docker.com/u/openbanking/**

The commits I made against the repository is available to view at: **https://github.com/OpenBankingUK/conformance-suite/commits?author=mbana**.

Tech: Go, Golang, Node.js, Vue.js, Vuex, Jest, Docker, Docker Compose, Kubernetes, OpenID Connect, JSON Web Token (JWT), Kompose, CircleCI, Swagger, WebSocket, Bitbucket Pipelines, OpenAPI 3.0, **https://bitbucket.org/openbankingteam/conformance-suite**, **https://hub.docker.com/u/openbanking/**.

Senior Software Engineer, 90POE - Ninety Percent of Everything Limited, London, UK - 02/2018-04/2018

I worked at startup specialising in software that runs on ship on two projects that were heavily Go-based.

platform-document-storage-service: Document storage and retrieval to be used by others services, so it's a core service. The core of the service was written in Go and exposed using gRPC and http using go gorilla/mux. Both write and retrieve supported arbitrarily large files which was achieved using gRPC unidirectional streams. The underlying store was MongoDB's GridsFS which I interfaced with using the Go driver mgo.

auditing: The service is structured very similar to the preceding in that the underlying service is exposed using gRPC but the top-level interface is done using GraphQL. I wrote the GraphQL server in go using graphql-go. The underlying store is in Postgres and the library I used to interact with it is GORM.

Tech: Go, Golang, gRPC, go gorilla/mux, Protocol Buffers, GORM, MongoDB, MongoDB GridsFS, mgo, GraphQL, graphql-go, Docker, Docker Compose, Kubernetes, NodeJS, Jest, Concourse CI, Postgres.

Full Stack Software Engineer, Root Capital LLP, London, UK - 10/2017-12/2017

I worked as a full stack Node.js software engineer on the Minds for Life application, mainly on the forum.

Frontend

- react, react-redux, react-boilerplate.
- Single Page Application (SPA) targeting mobile platforms.
- ES56 using most of the latest ES56 features; async, await, classes etc.
- Serveless and hosted on Amazon S3 as static assets, with Amazon CloudFront as the CDN.

Backend

- NodeJS server written in ES6, like the frontend.
- Koa.
- MySQL as the datastore, using the Knex.js library.
- Packaged as a set of docker containers.

CI, Devops and Infrastructure

- Services were packaged as containers. Used docker and docker-compose to start them.
- Builds managed by Semaphore CI and Wercker.

Tech: JavaScript, ES6, Node.js, react, react-redux, react-boilerplate, Webpack, Koa, Knex.js, Sequel Pro, Docker, Docker Compose, Kubernetes, Semaphore CI, Wercker, Amazon S3, Amazon CloudFront

Full Stack Software Engineer, Lloyds Banking Group PLC, London, UK - 03/2017-05/2017

NodeJS - JavaScript

- Loopback for server-side of the code.
- ES5/6-based code base.

Monitoring/Devops/Misc

- Splunk and sending logs via (**https://en.wikipedia.org/wiki/Syslog**) a LogDrain service available on Bluemix.
- Gerrit for managing code. CI:
- Jenkins: Configuring, managing and installing.
- Jenkins 2: Same as previous plus writing pipeline scripts.

Tech: JavaScript, Node.js, ES5/6-based code base, LoopBack for server-side of the code, SysLog, Splunk, Jenkins, Jenkins 2, Gerrit Code Review, IBM Bluemix

Senior Front-End Software Engineer, Synthace Ltd., London, UK - 04/2016-11/2016

Did a fair amount of architectural UI work:

- JWT-based authentication: Implemented most, if not all, of the authentication related UI features. Polymer didn't have an authentication module as it's fairly new requiring me to re-implement this feature.
- API interactions: I introduced Swagger JS and did the conversion from plain XHR to Promises, and ensured API was in-sync with the state of the authentication.
- Updates via the Web Socket for notifications and async task updates: STOMP Over WebSocket.
- Bootstrapped the testing using Web Component Tester.
- Deciding on the build, test and hosting strategy, e.g., hosting our own CDN using Azure.
- Performance: 1) pushed to have HTTP/2 enabled, and prototyped, on our custom server written in Go, 2) implemented lazy-loading of our Web Components which are included using HTML Imports, 3) Significantly improved UI build scripts; went from a somewhat un-deterministic build to one that almost always runs.
- Introduced ES6 to the code-base, and moving to defining Polymer elements using ES6 classes.
- Misc: libraries/utils to ease UI development.

We deploy Docker images to our Kubernetes cluster running in Azure using:

- Docker: Fairly comfortable using this.
- Kubernetes: I've done deployments of dev branches, so I understand the deployment model, navigating the Kubernetes dashboard and crude command line interactions, e.g., port-forwarding of the service the pod is running from the cluster to the local machine.

Added support our language called Antha to Monaco Editor, the editor that powers Visual Studio Code.

Since this is a startup I have done a fair amount of work and I have been given a fair share of responsibility, more so than any of my prior roles.

Tech: JavaScript, ES6, Polymer Project, Web Components, TypeScript, VSCode, Azure, Docker, Kubernetes, Go, Polymer Web Component Tester, Git, Swagger.

Software Engineer, IG Index Ltd. London, UK - 09-2014-02/2016

Price & Indicator Alerts My main responsibility was handling the UI aspect-setting, configuring, triggering-of the various Alerts we support, from the basic Price Alert to Technical Indicators such as RSI and Moving Average. **Tech New:** Modern UI powered by ES2015 (Babel), Ember.js, Handlebars and Less. The UI was then composed of isolated and reusable Ember components. Runs on a NodeJS server, managed with npm and bower, and version controlled using Git. Integration and unit tests written in Mocha then run using Karma. **Tech Old:** Vanilla JavaScript using an in-house framework when changing the previous UI.

Charts Assisted in the conversion of the Adobe Flex Real Time Charts to an SVG-based version. **Tech:** d3 and tested like above.

Deeplinking A hashed action link, like typical deeplinks, that we send to our Clients which then launches the mobile IG app, or directs them to the app store for the device with the IG app pre-selected. Upon login the action is carried out automatically. I did the bulk of the work with the team lead overseeing it. **Tech:** Java 8, Spring and acceptance tested using Cucumber. Redirecting and launching of the IG app was done using vanilla JavaScript.

Software Engineer, ITRS (International Trading Room Software) Group Ltd. London, UK - 02-2010-09-2014

JavaScript (UI) Software Engineer - 01/2014-09/2014 UI for the next generation of the product which is built around, loosely speaking, a real-time distributed analytics store. The aim is for the old system to stream data to the new system so we can provide all the great visualizations available in the HTML ecosystem, which was not achievable in a reasonable amount of time in Swing.

The code is entirely modularized using RequireJS so we can test each viewmodel without creating a view, we then test the entire UI (end-to-end tests) using WebDriverJS.

Some Java/C coding required to write NodeJS bindings to interact with the store. We evaluated several frameworks, Angular, Batman and KnockoutJS, by writing prototype applications that connected to our backend for a period of roughly 4-6 months before we chose to settle on Knockout.js.

Tech: JavaScript, NodeJS, Node-WebKit, Durandal, KnockoutJS, RequireJS, Git, Jasmine, Protractor (WebDriverJS), Jenkins, Bower, HTML5, CSS, LoopBack.io.

Java (UI) Software Engineer - 07/2012-12/2013 Worked with two software engineers and one QA member on a new Swing UI, ACLite, that uses our new streaming-based API to access Geneos data, for more info. see **https://resources.itrsgroup.com/OpenAccess**. We access data from a fault-tolerant Cluster that is built on a set of distributed nodes. The system we coded against is somewhat similar to the Amazon distributed key/value store, DynamoDB, except with support for streaming, so I am familiar with dealing with distributed systems.

Tech: Java, Maven, Swing, Eclipse, Jenkins, Git, Vagrant.

C++ Software Engineer - 02/2010-06/2012 Spent one year with Run The Business (RTB) team, a team set-up to fix critical bugs that Customers encounter. A very challenging role which requires all-around product knowledge, good debugging skills and being able to liaise with our Support staff in dealing with the Customers. Prior to this I was one of three software engineers working in the Transactions and Latency Monitoring team (part of the backend team) doing core C++. We wrote and maintained the following plug-ins that are part of the Geneos suite:

1. Geneos FIX plug-in: Monitors FIX (protocol) messages.
2. Feed Latency Monitoring Plug-In: Monitors feeds and calculates latency of instruments and fields across the monitored feeds.
3. Latency Monitoring - Message Tracker FIX adapter: Tracks, generally, FIX messages across several checkpoints.
4. Market Data Monitor: See below. And bespoke plug-ins written for specific firms.

We also maintained several non-finance specific plug-ins. I ported another bespoke plug-in called Price Latency Monitor (provides latency figures for bonds) to MS VC++ when I worked on this team. Projects:

- I converted the Windows version of the entire product suite from Visual Studio 2005 to 2010.
- I wrote the Market Data Reliability plug-in. This plug-in connects to the Patsystem's Trading API to monitor commodity prices, using their C API, to determine if prices are 'stale'.
- I ported a significant part of our product to Solaris x86-64 (64-bit non-sparc architecture).

Tech: C++, STL, Boost, Visual Studio, Linux/Unix, GDB, DBX, Make, Configure, XML, XPath, CPPUnit, Financial Information eXchange (FIX) protocol, Geneos FIX plug-in, Feed Latency Monitoring Plug-In, Latency Monitoring - Message Tracker FIX adapter, Market Data Monitor, Geneos PATS-STATUS Plug-In, Patsystem's Trading API.

Software Engineer Intern, then Tester, Thomson Reuters. London, UK - 05-2009-11-2009

C# Software Engineer One of four software engineers working on a search and navigation interface to Global Product Search. Full life-cycle of development; requirements engineering, analysis and design to implementation, testing and deployment.

Tech: C#, Silverlight 3.0, MS SQL Server 2005, LINQ, Web Services (WCF), XML and Visual Studio 2008. I handled deployment using CruiseControl.NET.

Tester User Acceptance Testing of the latest release of Thomson Reuter's 3000 Extra, then called UTAH, now called Eikon. UTAH combines the data from Thomson and Reuters. My primary responsibilities were to validate the end product against pre-defined requirements/workflows. 1. Worked on Thomson Reuters project UTAH as part of a large team. 2. Tasks included testing, observing, documenting software bugs, issues and errors before final release of Utah. 3. Testing was done over multiple iterations.

Education

- **2005-2008:** BSc Computer Science, City, University of London.
- **2008-2009:** MSc Software Systems Engineering, University College London, and Trading & Financial Market Structure module, London Business School.

Additional Information

Misc

- **Passport/Nationality:** I am a British citizen with a British passport.
- **Drivers Licence:** Full UK Driving Licence.

Languages

- **English and Swahili:** Native.
- **Arabic:** Intermediate. I have lived in Marrakech, Morocco for almost two years. I have also lived in Cairo, Egypt and have travelled several times to the UAE.