Michał Bultrowicz

Curriculum Vitae

Contact

michal.bultrowicz@gmail.com
(+48) 790 467 660
bultrowicz.com
Timezone: CET

Summary

A holistic back-end engineer. Using Python, containers, IaC, some HTML/CSS/JS, bits of Data Science, and anything else that's needed.

I can negotiate requirements, design, plan the delivery, implement, ensure quality, and setup monitoring.

I'm used to creating business-critical applications and handling the challenges and pressure that come with that.

I have a strong belief in gradual change, continuous testing, and data-driven decision-making.

In software I love simplicity, elegance, and resilience.

Computer skills

- expert in Python programming (including async)
- writing understandable and testable code
- building highly available (HA) and fault-tolerant back-ends
- advanced web application testing
- site reliability engineering - ensuring observability, gathering metrics, setting up alerting, optimizing performance
- cloud infrastructure automation
- ensuring web security (X.509 certificates, mTLS, JWT/JWK, OAuth2)
- implementing Continuous Delivery pipelines
- optimizing team's development workflows
- exposure to building front-ends with ReactJS and plain HTML+JS
- effective on the command line / in the terminal
- driving "spikes" (in the eXtreme Programming sense) through the entire tech stack, hardware-to-frontend

---

Tools (with years of experience)

- languages: Python (10), Bash (11), JavaScript (1.5), Java (5), C++ (1), C# (4)
- clouds: AWS (3), GCP (1), Digital Ocean (2)
- web APIs / REST: FastAPI (0.5), AioHTTP (1.5), Falcon (0.5), Flask (2), Django (3), Spring Boot (1.5)
- testing: pytest (7), Python's unittest (2), selenium (1.5), Mountebank / mountepy (3.5), Playwright (some experiments)
- SQL: PostgreSQL (6), Citus (1), AWS Aurora PostgreSQL (1), AWS Redshift (0.5)
- data stores: Redis(3.5), Elasticsearch(2.5), S3/MinIO (3.5)
- Kafka (4.5 total):
  - Confluent Kafka (2)
  - AWS Kinesis (1.5)
  - AWS MSK (1)
  - Faust (1)
  - aiokafka (1)
- containers: Docker (7), Kubernetes (2), Helm (1.5), AWS ECS (2.5)
- infrastructure management: Terraform (3), Ansible (2)
- version control: Git (10)
- logging/monitoring: AWS CloudWatch (1.5), DataDog (1), Prometheus (1.5), ELK stack(2.5)
- CI/CD: Gitlab CD (2), Github Actions (1), Jenkins (4), TeamCity (2)
- error tracking: Sentry (4.5)
- serverless: AWS lambda (1.5)
- operating systems: Linux(12), Windows (5)
- front-end: HTML/CSS (1.5), JS (1.5), ReactJS (1)
- data analysis: Pandas (0.5), Seaborn (0.5)
- GUI app development: Android (3), WPF(2), Java Swing (2)
- miscellaneous:
  - Celery (2)
  - nginx (3)
  - Hadoop MapReduce (0.5)
  - OpenSSL (1)

Soft skills

- leading a Scrum or Kanban team
- teaching developers through meticulous code reviews
- knowledge of software legal compliance and licensing issues
- analyzing teams' incentives and the tensions arising from them

---

Work experience

2023.07 - onward

Senior Python Developer at Scalo

Remote

The project (greenfield) is a financial data pipeline running on Azure cloud.

I'm working on it since day one, writing the code and helping the project's architect with the system design and technology selection.

Technologies used:

- FastAPI
- PostgreSQL
- Microsoft SQL Server
- SQLAlchemy
- Polars

2023.04 - 2023.05

Self-employed at WitchSoft
Remote

I created PITowalut.pl, a website that helps with calculations for Polish PIT-38 tax from transactions on foreign exchanges.

The entire transaction with the customer takes place on a single page with only a single redirect for the payment. The payment processor used is Przelewy24. The front-end's made with VanillaJS and PicoCSS, served from CloudFlare Pages. It has passwordless authentication with a code sent over e-mail. E-mails are sent with automation using a Google Workspace service account. The back-end is hosted on Digital Ocean. PostgreSQL serves as the data layer. The web API is built with async Python, FastAPI, and SQLAlchemy.

2022.01 - 2022.12

Senior Back-end Engineer at Nira
Remote

Nira is a real-time access-control system. It analyzes who has access to what documents in an organization's cloud, and allows to perform automated bulk changes to the access permissions. It integrated with Google Cloud, but we were working on adding support for more systems (e.g. Microsoft).

The team was 100% remote and globally-distributed, favoring asynchronous forms of communication, and promoting a high degree of autonomy. There were up to 25 engineers on it. Safety and confidentiality of users' data was paramount - we were HIPAA-compliant.

My main role was the development, maintenance, and ensuring reliability of Python back-end microservices acting as our asynchronous task queue - the "Actions" service. The service allowed to execute policy-driven and on-demand changes to document permissions via Google Workspace APIs.

My role included:

- technical leadership of the "Actions" service code and cloud infrastructure
- analyzing and improving back-end performance
- operational support of "Actions"
- Analysis and solution design for a variety of new feature bundles, which had "Actions" dependencies; working from product requirements to produce engineering plans and effort/complexity estimates, which flowed through into implementation projects

- Review of other engineer’s code as a part of quality assurance and release processes, both for "Actions" and the wider back-end ecosystem
- Revising and extending CI/CD pipelines based on GitHub Actions

Notable achievements:

- taking over the "Actions" codebase in 9 days, as the previous maintainer was leaving the company shortly after I joined
- introducing test harnesses, and performing a gradual redesign of the "Actions" codebase that improved both reliability and performance, all the while delivering new features
- "Actions" was capable of conducting bulk change operations for around 10 million documents
- switched out the old implementation based on AWS SQS and synchronous Python to one based on AWS MSK (Kafka) and asynchronous Python (with aiokafka)
- made it so that "Actions" microservices could be brought up locally (on the developers computer) with a single "make" command after git-cloning the repository

2021.01 - 2022.01

Working on my own software, cryptocurrency trading, travel

I was trading cryptocurrencies to get income while I was traveling with my wife, and working on my own software. The latter included (but was not limited to):

- a multi-platform desktop Kivy app that I intend to productize in the future
- an app to keep track of and plan my finances and investments
- scripts for setting up my OS, and keeping it updated and synced across computers

2018.11 - 2021.01

Senior DevOps Engineer at British Telecom (BT)
Remote, with occasional meetings in the UK

Main project: Vena

I worked on a team of external contractors tasked with developing a web application for designing the telecom infrastructure and the initial configuration of Cisco and Juniper routers. The network created by these routers would become the next iteration of British Telecom's core network in the UK. To do that we had to work intensely with BT engineers from various teams, piecing together the knowledge about company's current processes, and helping design the requested solution, that was supposed to fit into a grander project.

We were all supposed to be universal developers, writing code, preparing its deployments and interacting with any hardware necessary. We didn't have a traditional team leader, which was an interesting exercise in self-organization. My application team had 4 to 8 persons (it was growing), but the entire project had many teams.

The whole project was also part of the organization's agile transformation, with our contractor team leading the way.

Prominent technologies used:

- Django and Django REST Framework - most of our back-end and front-end
- PostgreSQL - our main database
- Kubernetes and Helm - deployments

- Nginx - reverse proxy
- Docker Compose - running a local, development instance of the app with its dependencies
- Minio - serving our web app's static assets, and storing media files
- Celery - running asynchronous graphs of tasks (with the "canvas" feature)
- ReactJS - a few more dynamic front-end pages
- Faust - reading streams of events from other systems from Kafka
- Redis - caching, Celery broker
- Sentry - error tracking
- Prometheus - metrics
- Junos PyEZ - managing Juniper routers
- netmiko - managing Cisco routers
- GitLab - code repository / continuous delivery pipelines
- Factory Boy - easy creation of ORM objects for tests
- terminal servers - logging in over SSH and getting a serial port terminal to the routers

My niches in the team were:

- code and infrastructure design for new features
- setting up automated tests and speeding them up
- refactoring for testability
- thorough code reviews
- teaching advanced Python and testable design to other team-members
- advocating for simple and tried OpenSource solutions we could control over heavier enterprise ones there wasn't enough team expertise in
- taking care of deployments and infrastructure (along with one other team member)

2017.05 - 2018.09

Python Back-end Developer at Iterio Data
Remote

My main project was a data pipeline for advertisement campaign and site traffic analysis. I worked as the main developer and the de facto architect of the system. Later on I also took over the maintenance and design of our cloud infrastructure. Other than that I did some ETL, the occasional simple data analysis, and polished up some old automations.

An important thing I learned here is that with the right tools and the right engineers, a small team (4, in this case) can create a web application processing the data of millions of people.

My duties included:

- creating back-end APIs with asynchronous Python (AioHTTP) and stuffing them in Docker containers
- designing our systems on AWS (ECS, Lambda, Kinesis, S3, CloudWatch)
- encoding our infrastructure and doing deployments with Terraform
- creating test strategies and tools for our cloud applications (including chaos testing)
- maintaining the quality of our services by doing stress tests and performance testing / monitoring
- helping in the design of our CitusDB (distributed PostgreSQL) schema

- setting up logging (CloudWatch, Google Stackdriver), metrics, and alerts (Sentry, PagerDuty)
- working out our development practices (trunk-based development, zero down-time deployments, etc.)
- maintaining old scripts that used enterprise SOAP web services (introduced Zeep to make that easier)

We worked like start-ups do, so there was context-switching and were always time-starved. Because of that the code I wrote had to be immediately understandable for me and my boss, who isn't a senior Python developer and was even more time-starved than me. He wanted to jump in real quick and tweak a few things from time to time, so extending the code had to be obvious as well.

Another hard job was picking what to test and how to do it. Tests were meant to actually save us time by not having to "deploy and poke around to see if everything seemed fine" after a change. We sometimes invested a lot of time in working out testing techniques for areas like AWS Lambda and microservices, because there doesn't seem to be enough literature on the topic. But sometimes there was just not enough time to test everything. Luckily, some parts of the system weren't user-facing, and putting them offline for a couple of minutes wasn't an issue (since we had queues, payload backups, etc.). So if anything went wrong while introducing changes to them, we'd be notified by our alerting system, and immediately implement fixes, without the users noticing any problems.

2017.05 - 2017.10

Ansible contractor at ScyllaDB
Remote

I was contracted to develop Ansible scripts for installation of ScyllaDB database. I managed to create automation for single instance installation. Later I started working on scripts for the set up of an entire cluster, but I didn't finish them. I was too overworked with this contract and a full-time job at Iterio, so I canceled the former.

The challenge here was mainly about having to base my work on existing complex shell scripts. They did things like modifying the kernel settings for maximized performance. The problem was that they were usually non-idempotent, and they were prone to modifying the machine's state. So I had to propose changes to them and recombine their parts into the Ansible scripts.

2017.01 - 2017.06

Software contractor at Techno Service SA / Siled
Remote, with a few factory visits in Gdańsk

The company had a multi-step manual process for creating PCB solder joint quality reports. The initial board analysis was performed by specialized industrial machines. I was hired to automate the process.

The old process looked like this:

- a person goes to the machine with a pendrive
- they run a software tool on the machine to download a set of testing results
- they copy the data to their computer
- they run a script that transforms the data into an Excel spreadsheet
- they send that spreadsheet to a person preparing a final report
- that last person manually copies fields from the sent spreadsheet into the final report

I created a desktop application through which you could specify batches of PCBs you were interested in, fill out some info, and immediately get the final report spreadsheet. That was possible because the PCB testing machine was connected to the network, and was running an MS SQL Server instance with all the board test data.

Technologies used:

- Python 3.6
- Tkinter, chosen because the GUI was very minimal and the application had to be very easy to install on Windows
- openpyxl for reading and writing Excel files
- PyAutoGUI for testing the GUI
- VirtualBox running Windows XP and Microsoft SQL Server 2005 - used that to test data extraction from the PCB testing machine (it was running that version of SQL Server)

2014.03 - 2016.04

Software Applications Developer at Intel Technology Poland
Gdańsk

Examples of assignments on this position, from the latest:

Leading a development team

I lead one of the teams that worked on Trusted Analytics Platform (TAP) project. TAP is a PaaS platform with main focus on data analytics. My team mainly did the integration of other teams' work, we overseen builds, deployments, releases and utility tools, but we also developed some back-end microservices.

I was responsible for planning, estimating and coordinating work, also teaching and mentoring of other team members. Furthermore I:

- acted as Python expert for all project teams
- created **mountepy** - a Python library to aid isolated microservice tests and increase our code quality
- influenced development procedures for all teams
- enforced software legal compliance and communicated with lawyers

Back-end microservices' development for TAP

I developed a few Spring Boot (Java) microservices and created one in Flask (Python). It was a data set indexing and search service backed by ElasticSearch. We used Cloud Foundry PaaS as a base for our applications. Sometimes there was a need to fiddle with virtual machines on which Cloud Foundry cluster was based on. I dabbled in load testing with Gatling (Scala) and Locust (Python).

Maintenance of a secure back-end for remote firmware updates (Intel Upgrade Service)

We had a mature distributed system, encompassing C# back-end services, ASP.NET front-end, a C++ server and a C++ client. The whole system was based on EPID digital signature scheme. I had to deliver patches to almost all of the application-layer components and do deployments to a large Windows-based infrastructure spread out geographically.

I also needed to run tests that touched everything from firmware on a client machine, through its drivers and application level software up to multi-tier back-end services and ending at server HSMs (Hardware Security Modules).

Implementing hardware-backed secure tunnels for certificate exchange

I needed to create a C++11 application for embedded Linux that could create, manage and transfer X.509 certificates. The certificates' private keys and the keys used to secure the connection to transfer them were created and stored in a hardware cryptography module (TPM). OpenSSL and TrouSerS were used for setup of MTLS (TLS with authentication of both parties) connections.

Researching using SIM cards as secure elements (SE)

Did that alone as my first task on the team. Had to look into Java Micro Edition and smart cards.

It turns out that some SIM cards (or smart cards with SIM card's for factor) are capable of asymmetric cryptography, but we couldn't use them easily. We would either need to get SIM operator rights to be able to push our code onto the cards with OTA, or we would need to get fresh custom cards from a smart card manufacturer. Both options weren't feasible for us.

2011.05 - 2014.02

Test Engineer Intern at Intel Technology Poland
Gdańsk

I maintained and developed a heterogeneous (Windows, Linux, Android) distributed framework for automated software, firmware and hardware testing. I've coded mainly in C# and Java (SE and Android versions), but I've helped myself with Python scripting.

My main focus was designing and implementing a versatile RPC-style communication between applications on different platforms (used WCF on Windows, JAX-WS on Linux, and my own protocol for Android). There was a tight collaboration with our clients - the validation teams.

Solving some problems required knowledge of obscure inner workings of .NET, Java and Windows system (e.g. discrepancies in implementations of TCP sockets on both programming platforms).

Notable talks

Developer workflow with local tests using Docker Compose
Pykonik Tech Talks #62, 2023-03-30. link

This talk features:

- Instructions on how to create functional and integrated tests using containers.
- A Docker Compose setup with the sample HTTP/REST app (written in Python) and a SQL database.
- Reloading the app container on code file changes.
- Running tests on code file changes.

- Makefiles to describe common development tasks.
- CI setup based on local tests.

TDD of Python microservices
EuroPython 2016 talk. link

Presentation of tools (some implemented by me) and solutions that enable Test-Driven Development of Python microservices. Told from the perspective of a maintainer of a single service.

Python microservices on PaaS done right
EuroPython 2015 talk. link

A collection of tips and practices that allow to have a successful microservices-based project. Concerns development, testing and work organization.

Education

2016.03 - 2016.05
Cryptography I course
Stanford (through Coursera)

2012 - 2015
Master of Science in Computer Science
Gdańsk University of Technology

Overall score: good plus

Specialized in Intelligent Interactive Systems (AI, computer vision, graphics)

Master thesis

Calling remote Java methods in Android system from .NET platform

Supervisor: Jacek Lebiedź, PhD MEng

The goal of this work was to create a set of libraries that would enable convenient remote code execution in Android system from .NET platform. The work was successful and had the side-effect of an upstream patch on `jsonrpc4j` library, enabling it to work on Android.

2008 - 2012
Bachelor of Science in Computer Science
Gdańsk University of Technology

Overall score: good plus

Final project

A C++ application using OpenCV that could learn and recognize faces using the eigenvectors method.

Languages

- Polish, native
- English, fluent

Interests

- technology and engineering in general
- psychology
- mythology (mainly Slavic, Nordic, and Greek)
- history
- self-optimization