Shako R. - Available from 06.09.21

Fullstack Python Software Engineer

Spacebus

Location: Azerbaijan, Baku (GMT +4)

Email: info@spacebus.dev

Soft Skills:

Highly committed & Self-motivated, People Manager and Communicator, Issues-solver, Team player.

Summary

Founder of MySQL User Group and Python User Group. True Pythonista. Actively sharing knowledge about Open Source. Currently managing AzePUG telegram group and Official Blog site. Author of several free Youtube courses.

In my spare time sending PRs to Open Source projects. Mosty doing code review style contributions. Please see my recent contributions here.
Author of MySQL-AutoXtrabackup tool.

[Part of previous experience]

Has OCP MySQL Administrator, RHCSA, online Python, and data engineer certificates from IBM.

Nominated as MySQL ACE and got the "MySQL 5.7 Community Contributor award 2015!"

Expertise

Full Stack Python/Django/FastAPI and NuxtJS + Golang. Deep Learning Engineering experience.

4+ QA Engineer in MySQL/Linux environment - automating tests for Backup solutions tools for MySQL.

Languages

- English (Upper, C1)
- Azerbaijani (Native)
- Russian (Native)

Technical Skills

Programming Languages:

- Python - CPython 5+ years
- Golang - less than a year
- JavaScript - as part of VueJS - less than a year.

Technologies and Frameworks:

- Django, Celery, Django REST Framework.
- FastAPI - Gino ORM.
- VueJS/NuxtJS - Vuex, API Integrations, user authentication with FireBase
- Serverless Framework(deploying AWS Python)

Databases:

- MySQL, PostgreSQL, MongoDB, DynamoDB

Containers

- Docker/Docker Compose
- Kubernetes - Google Cloud

Clouds:

- AWS - EC2, ECS, Serverless stack
  Route 53
- Google Cloud - MySQL database, APP Engine

CI/CD

- Gitlab CI + DigitalOcean
- Gitlab CI + AWS EC2 instances
- ArgoCD

Professional Experience

Work Experience:

Education

Innopolis University 2019

MSc Robotics

Azerbaijan State University of Economics 2007-2011

BSc Finance

Software Engineer June 2021 – Present
(SpaceBus - Ukraine)

- Doing project-based Python Backend services development.
- Involved in the Python interview processes.
- Focused on AWS solutions.

Fullstack Software Engineer Feb 2020 – April 2021
(Buglance - Azerbaijan)

- R&D team member.
- Doing project-based Golang & Python Backend services development.
- Using NuxtJS & VueJS as a frontend framework.
- Doing project-based Deep Learning research primarily focused on Computer Vision applications and their Web interaction.
- Using Python as a primary programming language.

Automation QA Engineer Jan 2019 – Feb 2020
(Galera Cluster - Finland)

- Using Python as a primary programming language for test automation.
- Continuously refine the QA process to improve product quality
- Troubleshoot and work with the development team to isolate issues
- Create and maintain automated tests
- Make Galera software builds (our build pipeline uses python, Jenkins, Docker, and Qemu)
- Assist customers with their questions and support tickets

Backup Solutions QA Engineer Oct 2015 – Jan 2019
(Percona LLC - USA)

- Using Python as a primary programming language for test automation.
- All aspects of backup solutions testing.
- Develop, automate QA strategies for testing.
- Evaluate new features, locate product issues and bugs.

Recently Completed Projects:

IMLA project:

- Description: the project is for collecting Azerbaijani Voice datasets from the Telegram bot by creating the contests and receiving voices as Telegram voice recordings.
- Collecting, crawling data from Azerbaijani internet space and from the books. Creating sentences from the grabbed data and storing them in the database.

- A telegram bot, statistics part was written in Golang and user authentication and sending the unique sentences part was in Python/Django
- During the contests in peak hours, users encountered some delays between showing the sentences, accepting the voices, and validating the voice data through ML API. We have solved it by caching the sentences to eliminate the duplicates and grab the sentences from the cache without the database hit
- Voice validation API was rewritten to be in an asynchronous way to not wait for the I/O cycle.
- During the high-load, we have migrated from a simple Docker environment to use auto-scaling with Kubernetes on Google Cloud(of course with the help of the DevOps guy)
- CI/CD pipeline was on GitlabCI and ArgoCD
- Optimized the data serialization part of Django REST Framework by eliminating redundant model serializers - manual serialization gave a performance boost.
- Optimized the Django ORM queries - spotted unindexed tables, heavy redundant database calls.

Used technologies: Django, Django REST Framework, Celery, Redis, PostgreSQL, Golang, AWS, Docker and VueJS

MILDA project:

- Description: The dating app for collecting extra real-life dating messages/sentences, preparing more real-life datasets.
- User authentication, APIs for managing photos, and all features were in Python/Django.
- The statistics part and CPU-intensive calculations were in Golang.
- Used AWS Lambda to crop, compress uploaded photos in the AWS S3.
- The photos and the user profiles were cached to get them fast and to eliminate duplication. The cache syncing was achieved by the background tasks.

- CI/CD part on GitlabCI again shifted to Kubernetes for auto-scaling features.
- Frontend pagination was used with prefetch feature up to 20 profiles not to send requests in each request.

  Used technologies: Django, Django REST Framework, Celery, Redis, PostgreSQL, AWS, Docker, Golang.

Face Detection/CV project:

- Description: The project aim is to create a simple, lightweight FD API and also streaming which will be used with any kind of CCTV camera.
- The requirements were: not to change any hardware in the already installed places and also to do FD inference through CPU not using GPU.
- With the ML engineer, we did a quick research and found the Intel OpenVino library which is suitable for CPU inference and designed in that manner.
- We have started from pre-trained state of art DL models and did the experimentations.
- The legacy code that we have found was written in Python and it was synchronous - so when you increase the connected camera count and try to read the frame - it was lagging.
- The biggest part is to rewrite the whole project to be asynchronous using asyncio. It was redesigned from scratch. I have learned a lot about asyncio during this process.
- Up to 50 cameras we have nearly eliminated the lag between reading, inference, and showing the frames to the user.
- The whole app was Dockerized to be easily installable to all platforms.

Used technologies: Intel OpenVino, Python/asyncio, Flask, Docker, docker-compose