**John de la Garza**
john@jjdev.com - (714) 895 6311 - Westminster, CA 92683

## SKILLS

- language: Python, Bash, C, Go, Perl, SQL, Lua, Javascript
- technology: HTTP, RDBMS (PostgreSQL, MySQL), DNS (BIND, Route53), Terraform, Linux (user and kernel), AWS (Lambda, ASG, ECS, ElasticCache (Redis), Kinesis, Security groups, etc)
- interviewing and mentoring
- programming

I love programming and building systems. Have worked on things from mircrocontrollers to distributed systems and everything in between. Discovering how things work is fun for me and will have no trouble with new hire learning curve. Prefer to work in a Unix like environment.

## WORK EXPERIENCE

### Blackberry, Irvine, CA, senior software engineer Apr 2019 to Sep 2023

- Was primary contact/maintainer of core system that supported a security product. Worked with small team to do a variety of tasks like developing new code, fixing bugs, deployments, sustaining, testing, and operations.
- We where the first team called when there was a customer facing issue. We would determine where in the system it was broken, who owned it, and most of the time fix it. This involved tracking down the cause in a distributed system with lots of moving parts and getting a fix deployed.
- Frequently pair programmed with others that where newer to the job to get them up to speed on using git consistently and learning how to run/develop/debug the different micro-services.
- Based on the fact that our core services where IO bound an asynchronous Python3 framework called 'Tornado' was used for many of the micro-services. Asyncio in Python is based on coroutines that cooperatively yield control (instead of blocking) while waiting on IO/blocking operations. This allows a single process to handle many requests concurrently. We had a backend of PostgreSQL (RDBMS) and Redis (caching and pub/sub).

  Other teams did things in Lua and Go. Which we eventually took ownership of.
- We became experts how things got deployed, on AWS services, and how to debug when there are many endpoints involved. A typical week could looking like working on a P0 production issue, adding new features, addressing bug reports, doing code review of patches submitted to the repos we maintained, and tuning (based on performance testing).
- One of the more interesting things was the use of openresty/lua (an app server that is forked from nginx). This system had a listening socket on the Internet that customer devices would connect to with a web-socket, the endpoint would use this bidirectional socket to send and receive. Incoming messages where routed to a set of Kinesis streams where Lambdas would pick them up and process them. While outgoing messages originating in a web app (for example) would get sent to one of the HTTP micro-services. This would result in a series of events that would ultimately publish the commands to a pub/sub message bus, where the openresty code would pick it up and send it down the socket.

### Comscore, Portland, OR Nov 2016 to Dec 2018

- Worked on a distributed system which was designed to count ads and determine who watched them. System consisted of retrievers to download data from clients, importers to normalize and store data, in house parallel database system for the bulk of the data, summarizers to

analyze the data, RDBMS for storing summary data for web front end, queueing subsystem, and exporters to transmit results back to clients.

- Created composable command line tools that acted as a language to assist in understanding incoming data and creating custom reports to be built combining these tools with some high level scripts.
- Participated in regular code reviews.
- Python, Perl, Bash, C, PostgreSQL, Linux

### Idealist.org, Portland, OR — Jul 2015 to Jul 2016

- Worked on a small team creating an HTTPS REST API that needed to inter-operate with an existing model, database, and messaging system. It provided HTTPS endpoints to allow other systems to post jobs in an automated fashion. This involved doing a deep dive into the existing code to fully understand the existing architecture.
- Was involved in the creation of a new system that would allow customers to bulk post job listings. It was available to the end users as an REST API where they could submit jobs, check status, etc. It was consistent, well documented, and designed to be discoverable.
- Wrote shell scripts to create chroot based sandbox. This provided a reproducible development environment on individual developer computers.
- There was a lot of collaboration. Code reviews where almost always done. We frequently pair programmed when it came to work on the more tricky parts. Work was done in two week sprints.
- Python, Flask, SQLAlchemy, PostgreSQL, Linux, RabbitMQ, Redis, AWS

### Rally, Wash, DC — Jun 2014 to Mar 2015

- Implemented a configuration system which used HTTP to internally publish configuration files that lived in a git repository. Pushing to the configuration repository triggered the new files to be published. This service was deployed on AWS using Chef.
- Created a program that would run on a cron job. It would pull down XML data, process, and insert into a local database.
- Python, Ruby, Linux, shell scripting

### TVplus, Orange, CA — Aug 2012 to Jan 2014

- Designed and implemented a read/write caching layer for with a HTTP REST front end. Remote data was represented by a graph and the local data was a set of trees that could be lazily loaded. and Flask.
- Designed and implemented a system to daily download XML data that represented TV program schedules for the next two weeks and merge/load it into a local database. This could mean scheduled shows where removed, added, or rescheduled. The system would try to reattach content for shows that where rescheduled (if a match could be guessed) and flag all other cases of disassociated content for human intervention.
- Created a service that took a URL to a video file and a description that specified how to edit the video. The goal was to mute audio and replace video with a black screen for the times and durations specified and then transcode to H.264/AAC.

  Once the new video file was created, a hash was generated and uploaded to an audio matching service. A request was sent to another server to have it generate a different type of hash. Once this was done the path to the second hash was added to a audio matching server.

  Most of this work was done in parallel and utilized multiple cores well.
- Setup development environment using LXC to allow a single server to run several systems to approximate our production environment which was spread across many amazon EC2 nodes.

- Python, SQLAlchemy, Flask, PostgreSQL, MongoDB, AWS, ffmpeg

Formal Education: Cal State Fullerton, BA Management Information Systems, 1997, Dean's List

Languages: English and Spanish