Written Interview for Python Engineer - Data Center Hardware Integration (Greater Boston Area)

Name Redacted

October 12, 2023

1 Career Development

1.1 How would you describe your level of experience managing data servers and network hardware?

I have an intermediate level of experience managing data servers and network hardware. I have performed most common tasks related[a] to configuring and maintaining data servers and network hardware at least once. In the past, I earned a Cisco Certified Entry Network Technician certification and am familiar with configuring hardware running Cisco IOS.

In both my graduate and undergraduate research labs, I received training in configuring and installing rack-mounted Linux servers. The training was very comprehensive, covering physical installations, account management, and software installation. I applied this skillset to install a handful of hardware, mostly GPU servers.

1.2 How would you describe your level of experience as a software engineer?

I currently have 3 years of experience building software as a machine learning engineer, researcher, and conultant. My prior experiences have honed my ability to ship maintainable code to production. I am comfortable writing code in multiple languages, with my strongest being Python. Here is an abbreviated list of professional software projects I have worked on in the past:

- Topic classification microservice for a microblog-based social media platform: Python
- Optimized post-processing pipeline for real-time object detection in video: Python, C++17, CUDA
- A cycle-approximate simulation of a RISC-V CPU down to the basic block level: Golang

1.3 What is your proudest success as an engineer?

To this day, my proudest success is the Discord bot which plays background music for my friends and while we hang out in voice chat. While it is not my most technically advanced creation, it is something I made which improves of the lives of people that are important to me. I often get requests for feature additions and bug reports from my friends even when I'm not online, which tells me that they use it even when I'm not around.

1.4 Outline the role of an engineering manager in shaping a high functioning team.

An engineering manager shapes a high functioning team in the following ways:

- **Hiring and Firing**: The Engineering Manager is the most aware of what skills would be needed to complement the existing team, and thus they should have a key role in selecting new hires for their team.

- **Managing Upwards**:The Engineering Manager should have the time to meet with leadership. This time should be used to ensure that their engineers have all the resources (e.g., time, information, infrastructure) they need to do well, and that expectations are managed accordingly.
- **Mentorship**: The Engineering Manager should serve as a "force multiplier" for the engineers they manage, by allocating work in such a way that the engineers can grow technically and professionally.

2 Experience

2.1 Describe your level of experience in Python, and how you have attained it.

The first time I used Python to create data visualizations for a research paper about 7 years ago, and it has been my favorite language ever since. Today, I'd I consider myself very experienced in Python. Here is a sample of work I've done:

- A Proof-of-Concept Flush+Reload attack on AES encryption, based on work by Yaroum and Falkner [8].
- A monitoring service written in Python that uses FFmpeg to check the integrity of 30-second videos being uploaded to S3-compatible cloud service.
- A text classification service for short-form blog posts, that is currently deployed in the production environment for a small social media app

I've also written blog posts where I've dissected Python bugs at the bytecode level, and maintained Python codebases that I haven't built/written myself.

2.2 Describe a case where it was very difficult to test code you were writing, but you found a reliable way to do it.

While working as a Machine Learning Engineer, it was common to encounter code which compiled and ran without software errors, but didn't perform as expected on certain edge cases. One of the first tasks I set about doing as a new hire was to develop a special test suite that would allow for us to measure whether changes to our image processing code affected the results of the object classification software (In terms of AUC ROC[4]). This way, every time a change was made to the image processing code for the classifiers, we could observe any potentialy degredation in our image detection capabilities and prevent it from being deployed.

2.3 When did you start working with Linux? Describe your level of experience as a user & developer on Linux.

I started using Linux on a regular basis 8 years ago for my computer science coursework and have been using it ever since. I've used a variety of Linux distros for software development, hosting, media playback, gaming, and more. Ubuntu was the Linux distribution of choice for the computing department at both my undergraduate and graduate institutions.

2.4 Describe your experience with networking, storage technologies and cloud infrastructure

I acquired professional experience with cloud infrastructure by working on a small engineering team at my previous employer. My role as a machine learning engineer involved the following:

- Writing Dockerfiles which assembled a custom SDK for `x64` and `aarch64` architectures
- Managing Artifacts in Google Cloud Platform Artifact Registry
- Writing custom CI workflows for GitHub CI and Jenkins

2.5 Describe your experience of large-scale physical server installations, including any provisioning, automation and service orchestration work.

My experience dealing with physical server installations is at the scale of adding, removing, and troubleshooting connections on one or two racks of servers. Notably, I collaborated with a professor to install several Lambda Labs Hyperplane 4-A100 servers into the rack at our university lab, connect them to the network, and create all of the user profiles.

When it comes to service orchestration, I configured Portainer to deploy and monitor our computer vision software on an array of several hundred x64 small form factor PCs. Portainer was connected to our cloud infrastructure, where we build container images for our software that Portainer could deploy to our hardware. We also configured log aggregation, downtime notifications, and enabled support for remote login when specific machines were malfunctioning.

2.6 Tell us what you believe teams should consider when they build, test, run and deliver software.

- Above all, adapt the process to the nature of the problem. The development cycle for a signal processing library on a 3-person team is different from infrastructure software at a 10,000-person company.
- Collect and evaluate requirements and their respective priorities early and often. They are the most useful guide for determining what needs to be built before delivery. They also determine how you build your test suite.
- Your first delivery should be when the subset requirements deemed critical are built and tested.
- A great way to stay on schedule and stay with the mantra of "ship early and ship often" is to always build according to the scale of your users.
- You shouldn't build a feature without a test plan written down somewhere.
- Every developer working on the product should have the means to build, test, and run their whole product without involving another human in the process.

Here are some texts that I cite as the basis for my beliefs:

- The Mythical Man-Month by Fred Brooks [2]
- The Joel Test: 12 Steps to Better Code by Joel Spolsky[9]
- Code Craft: The Practice of Writing Excellent Code by Pete Goodliffe[5]
- Game Engine Architecture by Jason Gregory, specifically the Chapter 3 of Part I [6]

2.7 How do you think about quality?

Quality is what you remember after using a product or service. It's what provides the sensation of "wow, this really just works" when using a piece of software. Humans are highly perceptive creatures, and thus high quality products and services require lots of attention to the finest details.

High quality software development follows a very similar pattern. The codebase is managed with a version control system. There should also be systems for automated building, testing, and delivery in place. If a new person joins the software development process, their ability to effectively contribute should be proportional to their related experience. The documentation for the software should comprehensively cover the problem(s) that the software intends to solve, and all of the details related to how the software solves them.

My experience with high quality software development comes from my time working as a consultant building natural language processing software. I was building a service that was incorporated into a codebase that was already many hundreds of thousands of lines, but was easy to digest when broken into small parts. It was easy to delineate what information I needed to know to solve the problem at hand, thanks to the thorough documentation and great communication skills of the engineers I was assisting as a consultant. Once my code was written and the service was deployed, it ran very efficiently on the hardware provisioned.

2.8 What would you like to achieve in career development and skills development?

- Build a strong network in the open source community
- Deliver excellent software that makes advanced technology more accessible to all
- Build a platform as a talented engineer who can both build great products and communicate effectively for a global audience.

3 Education

3.1 At high school, how did you fare in mathematics and physical sciences? Which were your strongest subjects in the hard sciences, and how did you rank in your class?

I was very strong in math and physical sciences in high school. I competed at the state level in both domains, and was ranked 18^{th} out of a class of 496 (Top 5%). Here is a sample of courses where I attained the highest mark:

- Advanced Placement Calculus BC
- Advanced Placement Computer Science
- Physics
- Chemistry
- Biology
- Algebra
- Geometry

3.2 At high school, what leadership roles did you take on?

Here is a list of relevant leadership roles from my high school years:

- I was the lead editor for all Math and computing content at my highschool's educational newsletter.
- I tutored math and Spanish for 4 hours a week as a volunteer.
- I represented my high school at the University of South Carolina Summer at Moore business competition.

3.3 What course and university did you choose, and why?

I attended Clemson University and acquired a Bachelors of Science in Computer Science, because understanding how computers work has been my passion since a very young age. I was awarded the Palmetto Fellows Scholarship (along with other awards), and thus was given the opportunity to attend at very low cost, cementing my choice of university.

3.4 How did you rank competitively in university? Which were your strongest courses, and which did you enjoy the most?

I was an A/B student in my undergraduate studies, with a cumulative GPA of 3.44/4.0. Here is a sample of courses where I recieved the highest grade in my undergraduate:

- Network Programming
- Design and Analysis of Algorithms
- Calculus of Several Variables

I really enjoyed courses that allowed me to take skills I had developed earlier on and build projects, here are some examples:

- 2-D Game Engine Construction
- Applied Data Science
- Robotics

My academic showing during my Master of Science in Computer Engineering was much stronger, with a GPA of 3.8/4.0.

3.5 At high school and university, describe your achievements that were considered exceptional by colleagues and staff.

- **Clemson School of Computing Best Research** I built monitoring infrastructure to trace memory accesses in Nvidia's Unified Virtual Memory system, and eventually presented the research at Super Computing 2019.
- **1st Place South Carolina Junior Academy of Sciences, Math and Computing** I studied reward allocation behavior in Bitcoin mining pools from to determine if certain policies were more beneficial to small, individual mining participants such as myself.
- **LQTLD3** I wrote a map-parsing library for a robotics project that allowed for the path-finding algorithm to determine available adjacent space in constant time, and then made it open source. My implementation is based on a paper by Kunio Aizawa and Shojiro[1] and the repository is hosted on GitHub.

4 Context

4.1 How are you involved in open source software? Describe any significant contributions to open source (with links where possible)

As a professional developer, I am a contributor to Akita[10], which is an MIT-licensed CPU/GPU simulation framework for building and testing microarchitectures against different workloads. I became a contributor for Akita through my graduate research, where I used Akita to build a RISC-V emulator.

As a user, I see it as my duty to file issues and work on improving the product when possible, however previous employers chose not allocate time towards merging our code upstream in cases where we did work with open source software.

4.2 What do you think are the key ingredients of a successful open source project?

- **Effective leadership**, who is capable of communicating priorities based on a strong vision, and running the project well at it's current stage. Effective leadership for open source manifests itself in the form of open governance and a diverse base of contributors.
- **Passionate builders**, who are technically capable to convert priorities into features. Often in the early stages, the leadership is also building. Passionate builders are going to be capable and willing to put in the extra time to create a high quality project.
- **Transparency**. Open source software wins because it is extensible and transparent. It is easiest to build on software when the person building on top of the software has all of the control they need to get the job done. There was a great essay on this recently by the Co-founder of Lago[3].

4.3 Why do you most want to work for Canonical?

I firmly believe in Canonical's mission to make cloud infrastructure open and accessible. I originally applied to graduate school so that I could help make advanced machine learning methods more accessible. Building infrastructure with Canonical is a solid step towards making my mission a reality.

4.4 Which other companies are building the sort of products you would like to work on?

Here's what I seek out in terms of projects:

- Technical Challenge to build / improve
- Provides utility to people
- Generally related to a subject I'm experienced with (e.g., Hardware/Software, Audio/Video, Machine Learning)

Here are some examples of companies that make products which meet those requirements:

- Apple (WebKit, Clang/LLVM)
- Microsoft (GitHub, Visual Studio Code)
- HuggingFace (HuggingFace Hub)

4.5 What do you think Canonical needs to improve in its engineering and products?

I think Canonical would benefit from a greater emphasis on "turnkey" workflows for certain classes of projects. I think an important test would be to create a non-trivial web service like the video streaming service described in this blog post[7] using Canonical's own managed service, and then see what it's like to build the same service using the offerings of the competition.

My opinion comes from positive experiences I've had with software services in the past: Namely, Heroku's famous `git push heroku master` workflow, as well as the workflow for setting up static websites on GitHub Pages. These workflows stick out in my mind as being extremely accessible, much like my first experience installing Ubuntu long ago.

Both of these examples cater to small developer shops and hobbyists who are price sensitive. If these markets aren't appealing to Canonical , then alternatively, Canonical could place a focus on consulting for the cybersecurity/privacy (SOC2 compliance, simplified IAM, HIPAA compliance) domain.

4.6 Who do you think are key competitors to Canonical? How do you think Canonical should plan to win that race?

The key competitors to Canonical are RedHat/IBM, SUSE, and HashiCorp. These companies compete with Canonical by providing software which attempts to simplify and streamline managing enterprise software + infrastructure. I think beating out this portion of the competition comes down to providing a superior UX on the handful of workflows that matter the most to a target userbase.

References

[1] Kunio Aizawa and Shojiro Tanaka. A constant-time algorithm for finding neighbors in quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1178–1183, 2009.

[2] Frederick Brooks Jr. *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley Professional, anniversary edition edition, 1995.

[3] Anh-Tho Chong. Open source does not win by being cheaper. URL: https://github.com/getlago/lago/wiki/Open-Source-does-not-win-by-being-cheaper, 2023.

[4] Google Machine Learning Education. Classification: Roc curve and auc. Foundational Courses: Crash Course, 2022. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[5] Pete Goodliffe. *Code Craft: The Practice of Writing Excellent Code*. No Starch Press, 2007.

[6] Jason Gregory. *Game Engine Architecture*. CRC Press, third edition, 2019.

[7] Alexandre Olive. Video streaming at scale with kubernetes and rabbitmq. URL: https://techblog.skeepers.io/video-streaming-at-scale-with-kubernetes-and-rabbitmq-6e23fd0e75fb, 2023.

[8] Derek Rodriguez. Flush reload attacks on aes. URL: https://github.com/dwrodri/flush-reload-aes, 2019.

[9] Joel Spolsky. The joel test: 12 steps to better code. URL: https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/, 2000.

[10] Yifan Sun. Akita. URL: https://github.com/sarchlab/akita, 2018.