Heinrich Curtis

Software Engineer





B.S. in Computer Science (Math minor)
Virginia Tech, 2021

https://github.com/Heinrich-Curtis/

None

(915) 449-7956

heinrich.curtis@smoothstack.com

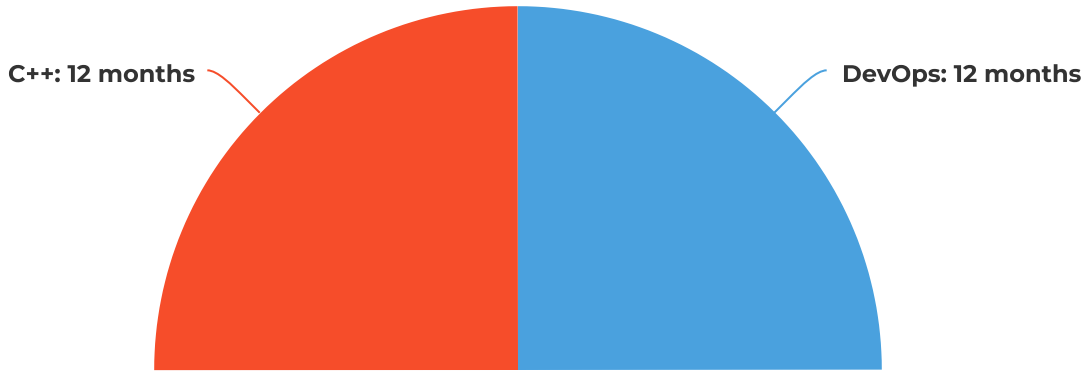
Skills Summary

Junior software developer with the knowledge and skills in a wide variety of domains to be a valuable member of any software development team.

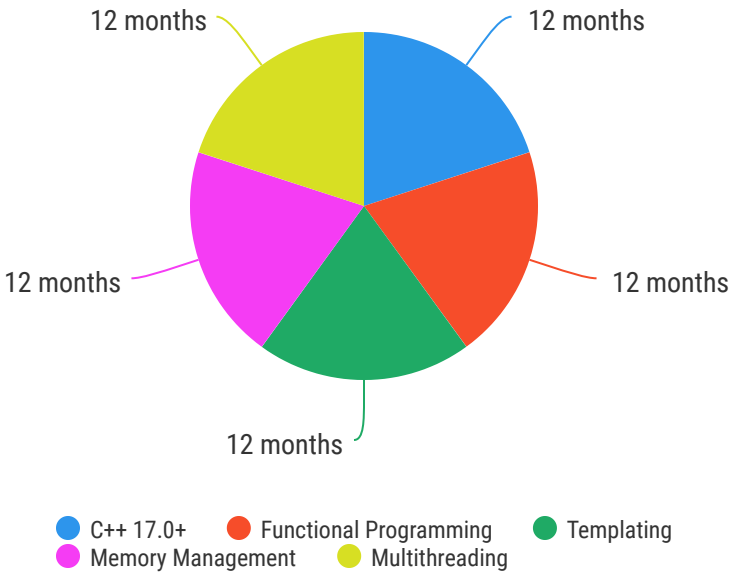
Technical Proficiencies: C++: C++ 11/14/17, g++/clang++, gdb, valgrind, DRD, Helgrind, STL, pthreads. Unix: Ubuntu 20.04, Centos 8, Fedora 35, make, cmake, package managers (yum/apt/dnf). Web: HTTP 1.1, TLS, OpenSSL, PKI, httpd. Cloud: git, Jenkins, Sonarqube, Docker, AWS (EC2, ECS, S3, CloudWatch, CloudFormation)

- Using data from the /proc filesystem and Unix tools to gauge application performance (top/htop/iotop/iftop) and tune application parameters (std::thread::hardware_concurrency)
- Command line development editors and tools for work in remote environments: sh/bash, make/cmake, nano/vim, bash scripting. Visual IDEs including VSCode, Eclipse, CodeBlocks
- Multiple compilers (gcc/g++, clang/clang++, msvc, minGW) and compiler options to diagnose errors during compilations: compiler errors (syntax errors, template deduction failures) and linker errors (undefined references, mis-configured library and include search paths)
- Rvalues and r-value references, as well as using them to implement move semantics to increase efficiency of working with types that support those features
- Writing parallel applications using facilities like thread and async, future and promise. Synchronization devices like mutexes, semaphores, barriers
- Docker containers: writing docker files, exposing networking ports, accessing services located on containers, managing and updating containers
- AWS services like EC2, S3, AIM, Security Groups, Autoscaling Groups, ECS
- Proficient with tools to troubleshoot malfunctioning programs: Valgrind (leak-check-full, DRD, Helgrind), gdb (breakpoints, backtrace, info threads)
- Knowledge of low-level networking concepts: the OSI model, network routing, network protocols, Unix sockets, TCP/IP
- Comfortable operating in a team environment and helping to meet organizational goals: attending and contributing in scheduled meetings, being available to assist co-workers, giving and receiving criticism, performing code reviews, establishing priority for upcoming work items

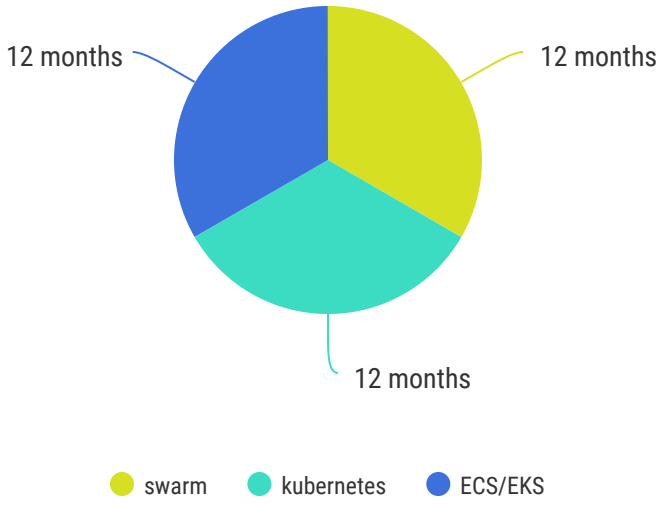
Industry Equivalence



C++ Technologies



Docker Technologies



Work Experience

Smoothstack Inc. McLean, VA **December 2021 - Present**
C++ Developer

Capstone Project Name - Utopia Server

The project goal was to take an existing HTTP server library and customize it, adding support for X.509 certificate signing via a certificate authority using 256-bit AES encryption.
The server is highly customizable, configurable either by way of a supplied configuration file or by way of Unix environment variables set at the time of server startup.
The live performance of the server is tracked using metrics from within the application itself as well as via metrics gleaned from Unix tools identified above (/proc, htop, etc.). Testing by adjusting the server configuration and work load was performed to identify the configuration that was most efficient.
Actions performed by the server and errors encountered at runtime are logged to a file along with the performance data. The log files will be pushed to an AWS S3 bucket where they can be accessed remotely, and eventually rotated into Glacier storage.

Responsibilities:

- Create and administer an X.509 certificate authority: protect the root certificate by dividing responsibility for user certificate signing between subordinate Cas (document signing vs. TLS), publish PEM/PFX bundles
- Validate input parameters to the server using shell scripts that use regular expressions and other logic (identifying the use of well-known ports) to ensure well-defined behavior at server startup
- Responsible for all logic used for logging: starting a logging thread, instrumenting the server code to log all events of interest, collecting data from the application and operating system used to perform create live performance data
- Responsible for all logic used for gathering performance data and analyzing/presenting it: using the output of Unix utilities and data from the logs to create metrics like the number of concurrent connections and the average throughput in requests per second, as well as parsing the logs to generate input for visualizations of the performance data
- Leveraged AWS to efficiently and securely host the server application, provide access to it from anywhere, store the logged data, regularly rotate the logs, and present requested logged data
- Create and maintain templates for Docker files and AWS CloudFormation that allow for rapid deployment of the application as well as scalability for future growth

Virginia Tech Transportation Institute **2017 - 2021**
Data Reductionist

- Analyze collected data for relevance to ongoing research projects according to specific project protocols, handle and safeguard confidential subject data

Parallel HTTP Server (C)

- Multi-threaded HTTP server capable of handling 10,000+ simultaneous connections
- Supports user authentication to the server via encrypted JSON Web Tokens Web Systems

V2X Radio Firmware and Message Processing (C on radio side, C++ on processor side)

- Modified firmware of a V2X radio to forward selected OTA messages over Ethernet to the processing computer mounted in the vehicle
- Wrote the classes for the received messages in the processor

Power-Based Load Balancing Cluster (Python)

- Django web app that receives requests for manipulations of user-submitted images and uses compute node power information to load-balance requests across workers
- Power and usage statistics collected from the Top Linux utility and Intel RAPL framework on each node
- Wrote all the logic for performing image manipulations using the ImageMagick Linux utility

Memory Manager (Java)

- Memory manager stores and returns records submitted from users
- Uses a “buddy-block” system to reduce internal/external memory fragmentation
- Utilizes a free list to reduce requests for memory from the OS