Ryan Parman

Cloud-native engineering leader with a focus on reliability, scalability, and security for the modern web.

Hi there 👋

Currently: Principal Software, Security, and Cloud Engineer leading the CloudOps and Engineering team at McGraw Hill.

Ryan has a passion for working on lower-level projects and pipelines which help make the lives of engineers easier, then making those projects usable and understandable to people. Some recent examples:

- Base disk OS images used in tens-of-thousands of nodes — both VMs and containers.
- Shifting monitoring and alerting left into reusable code used by hundreds of teams.
- Shifting infrastructure left with reusable Terraform modules created as building blocks that can be mixed and matched.
- Custom cybersecurity tools for scanning and reporting — including tracking the cert expirations on tens-of-thousands of endpoints on ±7,500 root domains and auto-alerting the teams not using Amazon Cert Manager or Let's Encrypt when certs are within a few weeks of expiring.
- Using off-the-shelf software like Artifactory, GitHub Enterprise, GitHub Actions, Circle CI, Jenkins, and more.
- Using highly-customized vendor-provided software like AWS Control Tower.
- Auto-rotating secrets with a *Token Vending Machine*.

Summary

Professional Blurb

Ryan Parman is a cloud-native engineering leader with a focus on reliability, scalability, and security for the modern web. As an engineering problem-solver with over 20 years of experience across software development, site reliability engineering, and cybersecurity, he understands how to listen, learn, adapt, and improve. He was a founding member of the AWS SDK team; patented multifactor-authentication-as-a-service at WePay; helped define the CI, CD, and SRE disciplines at McGraw Hill; came up with the idea of "serverless, event-driven, responsive functions in the cloud" while at Amazon Web Services in 2010 (AWS Lambda); and much, much more.

If we were having coffee...

I have been building things for the web since 1998. I've lived through the browser wars (both of them), I've worked on multiple high-profile projects, and have maintained server clusters powering hundreds of millions of dollars-worth of transactions. I have experience with startups, not-so-startups, Fortune 500s, and heavily-used open-source projects. I have lots of experience working across large distributed teams to get projects completed.

I have experience taking the long-view on things that people might not understand today. I understand that "perfect" is the enemy of "done", but conversely that "we must not ship crap." I understand that the "minimum viable product" version of a motorcycle isn't the chrome wheels or a nice chassis, but a tricycle (e.g., minimum *usable* product). I understand that it's easy to ship something, but hard to maintain it. You need diligence, focus, patience, and lots of really good cross-training and documentation to be successful.

When it comes to software, there is *no such thing* as "if it ain't broke, don't fix it." Software that is not maintained falls into bit-rot, and becomes more expensive to fix later than if you'd simply maintained it as you went along. Patching security vulnerabilities *is* an example of providing *direct customer value*. Just because the customer can't see it, doesn't mean it's not helping them.

I understand that we all rise and fall together, so I place emphasis on tearing down walls between departments or divisions so that we can work better together. My experience spans across UX, development, operations, cybersecurity, and documentation — as an individual contributor, an engineering manager, and a technical/thought leader. I excel in teams that care about the customer or end-user, and want to make things better tomorrow than they are today. I excel in teams where I am given the latitude to make decisions and work across teams to deliver the best possible customer experience. Let's work together to create something amazing!

As I've learned more and more about managing humans over the years — some from people, and some from books — much of my personal leadership style is inspired by these two excellent books: Managing Humans by Michael Lopp (aka *Rands*), and Trillion Dollar Coach by Eric Schmidt, Jonathan Rosenberg, and Alan Eagle from Google. I regularly repeat a few mantras that my teams get sick and tired of hearing me repeat over and over again, but I believe they are fundamental truths to being successful engineers:

- About automation, "Let the robots do what the robots are good at, so that humans can focus on doing the things that humans are good at."
- About people who are frustrating, "You can't change anybody else but yourself. Expecting other people to change just because you want them to is a recipe for perpetual disappointment. But you can choose to change how you look at this person/situation."
- About most things, "There are things that matter, and there are things that really, really don't. Focus on the things that actually matter, and stop wasting your time and energy on the things that don't."

Technical Skills and Software

While my experience and personal technical interests are broad, the following list is focused more on my interest in DevTools, DevOps, and SRE roles. I would be happy to share additional experience for other areas upon request.

> **NOTE:** I've seen so many résumés over the years as an interviewer that list a whole bunch of skills as though they are all equal. In truth, they almost never are, so I've tried to do better by adding an approximate proficiency level (scale: Low, Med, High, Expert), as well as a directional arrow. An up-arrow (↑) means I'm actively working with them and my proficiency is likely to go **up** over time. A down-

arrow (↓) means it's been a while since I've worked with it, and my proficiency is likely to go **down** over time unless I get a good refresher course. *No arrow* suggests that I simply maintain my knowledge where it is.

- **Operating Systems:** macOS (Expert: ↑), CentOS Linux (High: ↑), Amazon Linux (High: ↑), Alpine Linux (High: ↑), Windows (Med), Ubuntu Linux (Med).
- **Standard Software Engineering Toolbox:** OOP fundamentals, dependency injection, polymorphism, performance, character encodings, Git, Linux, Makefiles, `yum`, `brew`, and other fundamentals (High: ↑); memorized algorithms (Low); memorized Big-O notation (Low; I never learned it formally, and the *notation itself* has always been a lower priority than learning to *do the work* of being more efficient).
- **Programming Languages:** *Modern* PHP (Expert: ↓) (not the bad old PHP that everyone hates), Bash (High: ↑), Browser JavaScript (Medium: ↓), Node.js JavaScript (Medium: ↓), Golang (High: ↑), Python (High: ↑), Ruby (Low: ↓). Interested in learning Swift and Rust, but am just scratching the surface.
- **Cloud Computing:** Google Cloud's core infrastructure services (Med: ↓), AWS (EC2, RDS, S3, CloudFront, SQS, SNS, IAM, STS, CloudWatch Monitoring, CloudWatch Logs + Insights, Lambda, ECS-on-EC2, ECR, API Gateway, Auto-scaling, CloudTrail, Elastic Transcoder, ElastiCache, Route 53, ELB/ALB, ACM, SSM, Parameter Store) (mostly High/Expert: ↑), AWS SDKs + CLI (High: ↑), Docker (High: ↑).
- **Provisioning:** Terraform (Expert: ↑), Terragrunt (Med/High: ↑), Packer (High: ↑), Ansible, Vagrant (Med: ↓).
- **API & Scalable System Design:** Understanding and designing highly-scalable, distributed systems for running web applications and web services (High: ↑). ~~REST~~ JSON-over-HTTP web service API design (High). *True* Representational State Transfer and an architectural style for Distributed Hypermedia Systems (REST, HATEOAS) (Med/High: ↑). GraphQL (and N+1) implementations (Med/High: ↑). Understand the difference between a true service-oriented-architecture (SOA) micro-service vs a "distributed monolith" (High: ↑). OpenAPI (née Swagger) (Med: ↑). JSON Schema (High: ↑).
- **Enterprise Services:** Artifactory (Expert: ↑), Jira (High: ↑), Confluence (High: ↑), GitHub Enterprise (High: ↑), GitHub (High: ↑), Pingdom (Med: ↓), New Relic (Med: ↑), Datadog (Med: ↓), Papertrail (Med: ↓), Slack (High: ↑), PagerDuty (High: ↑).
- **Databases & Key-Value/Document stores:** MySQL (Med: ↑), Redis (High: ↓), PostgreSQL (Low: ↑), Memcache (Low: ↓).
- **Metadata and Config Formats:** RDFa, Dublin Core, FOAF, OpenSearch, JSON-LD, Microformats, RSS, Atom (RFC 4287), JSON, YAML, TOML, XML, HCL.

Work Experience & Notable Projects

McGraw Hill (née McGraw-Hill Education) — Remote (since COVID), previously Seattle, WA

Principal SRE and Cloud Engineer (June 2020—Present)

Continuing the work I led as an engineering manager, I migrated into a more strategic role around the projects where I had started as the creator, initiator, primary developer — planning the path of the products and how they wove into the larger tapestry of our highly-heterogenous application ecosystem which had grown by way of acquisition over the years. With no longer having direct reports, I was able to focus on *technical leadership* without the responsibility of *human management*.

- **Documentation:** Prolific documentarian. Documentation is worth 50% of your grade.
- **Reliability Platform:** Products that I had personally pioneered (ECS-optimized Base AMI, Prism, Monitoring-as-Code, Terraform modules) became core pieces of our "reliability platform" alongside off-the-shelf software/services such as AWS Control Tower, Artifactory, GitHub Enterprise, GitHub Actions, Circle CI Enterprise, Jenkins, and more.
- **Control Tower:** My team and I partnered with McGraw Hill Enterprise Architecture and AWS Professional Services to deploy AWS Control Tower and AWS SSO. Dramatically lowered costs and increased control over account guardrails. Enabled automated provisioning of new accounts (i.e., "the Account Factory"), and developed smoke tests as a post-provisioning validation step.
- **Clarity in Complexity:** Collaborated on the Guardrails (mandatory + custom) deployed across all AWS account *organizational units* (OUs). These were written as CloudFormation YAML, Python, and Bash scripts. In such a large complex, project, it's easy for the code to become obtuse and difficult to trace. Worked with my team to make sure we understood the fine details of the implementation, then implemented Lambda functions and CI code to read certain changes in Git commits to master/main and generate README/Confluence documentation with directed graphs and charts generated from DOT documents, to make the workflows and details easier to understand visually.
- **Base AMI program** (ECS-optimized, General Purpose Linux, General Purpose Windows Server, and *derivative* AMIs for things like Artifactory and GitHub Actions). Took what we'd learned about Packer, CIS Benchmarks, security patching, and the needs of a particular AMI's audience to develop a single build pipeline which brought the best ideas from each AMI together — automatic dev builds with unit/integration testing on Git commit, production builds with complete package indexing on Git tag, pre-installing and pre-configuring agents for metrics and cybersecurity, automated security analysis scanning, making the Base AMIs available to all ±150 AWS accounts, rotating the hosts to use the new AMI with zero downtime. Adopted EC2 ImageBuilder in the process.
- **Streamlining:** Combined elements of Terraform, Monitoring-as-Code, Base AMIs, and our custom security tooling to empower application teams to bring a Docker image with a small amount of configuration and deploy it to one of our Amazon ECS clusters with best practices, infrastructure monitoring, and operational tooling built-in, lowering overall costs.

- **Preventative automation:** Scanned Route 53 and other DNS providers to obtain a mapping of our 1000s of active websites. Leveraged highly-concurrent, scalable bots to fetch certificate data from each endpoint. Enabling faster rotation for expiring datacenter certs by knowing both WHICH certs and WHERE they were installed. Verified the required DNS records for self-rotating Amazon Cert Manager certs.

- **Prism:** Developed custom security and operational tooling where off-the-shelf tools wouldn't give us what we needed. Solution involved highly concurrent and dynamically-scalable nodes that would scan the AWS APIs to understand the current posture of ±150 AWS accounts. Made the data transparent to ALL engineers, enabling teams to be involved in improving their infrastructure stacks.

- **Automation for Artifactory:** Rebuilt our Artifactory cluster with a "cattle, not pets" approach. Dedicated Base AMI, rotated monthly. Migrated artifacts from NFS to S3. Rewrote configuration in Terraform instead of by-hand. Moved service-user management into Terraform. This automation reduced the amount of human error in the process, and improved our security posture.

- **Docker Image + Custom Packages:** Worked to streamline the developer experience by moving all disparate Amazon ECR Docker image repositories into Artifactory. Worked to reduce the time to build VMs and Docker images by identifying the common software people were manually installing, and began packaging them as pre-compiled `.rpm`, `.deb`, and `.apk` (Alpine Linux) packages that could be installed from Artifactory through the system's built-in package management system. Faster builds with better reliability and reduction of the "left-pad" problem.

- **Token Vending Machine:** Built a Token Vending Machine to enable continuous token/password rotation for our engineering teams. "Push button, receive token." Solution leveraged Secrets Manager, Lambda, KMS, IAM policies, and some custom CLI software written in Go. First integration was for service-users (robots) in Artifactory.

- **Training and Education:** Worked to develop the SysAdmin "button pushers" on my teams into more well-rounded software engineers who could automate more reliabily. Continued to push to *raise the bar* in the quality of our team. When SysAdmins left the company, worked to hire *true* SREs to fill their spots.

Engineering Manager, Site Reliability (October 2018—June 2020)

Owned, and was the key decision-maker for the development of a core platform of company-wide, reliability-oriented projects. With our development teams moving toward Full-Cycle Development, our SRE team focused on solving more macro-oriented problems which affected more than 75 decentralized engineering teams across the company. These projects have empowered greater self-service for engineering teams, enabling them to move faster without having to reinvent the wheel.

Many of the following projects got their start in my work as an application engineer for MHE, and carried over into this role.

- **ECS-optimized Amazon Linux Base AMI** for all Amazon ECS applications. Modified the version

vended by AWS to meet Level-2 CIS Guidelines for both Amazon Linux and Docker. Underwent deep collaboration with security, operations, and various business units to ensure strict compliance with requirements. Achieved high levels of opt-in adoption, which gave security and operations orgs higher levels of confidence in the product development teams.

- **Prism** which is an "executive dashboard" enabling significantly improved visibility into the security and operational configurations of our AWS accounts (several dozen). Enables visibility to Engineering Managers, Directors, VPs, and the CTO, while also providing clear instructions to app engineers about why the configuration is incorrect and what needs to be done to resolve the issue.

- **Monitoring-as-Code** which leverages Terraform and Python to streamline the process of generating and maintaining dashboards and monitors in Datadog and New Relic across a large, heterogeneous swath of applications. Trained development teams in adopting "full-cycle" development practices where the development team owns day-to-day operations of their services including deployments, support, and on-call rotations.

- Formed a leadership group to develop a more rigorous process for developing, patching, vending, and maintaining re-usable **Terraform modules** that are used by large numbers of product development teams across the company. Standardized their development, contribution, and usage guidelines, adopted an Apache-style "incubator" for developing new modules, and adopted a process for shipping LTS-style packages of modules.

- Took over engineering management responsibilities for the **Site Reliability** group in MHE's Seattle office. Worked to integrate our office better with the larger, developing SRE practice across all offices. Joined the SRE leadership group to help guide and participate in the development of better processes around reliability, which we then worked with product development teams to adopt and apply.

- Rebooted our Seattle SRE **interview process**, with a much higher focus on identifying high-quality engineers with a 70/30 split between software engineering (Dev) and systems engineering (Ops), and who were more *leaders* than not. Integrated many ideas and *leadership principles* from my time working at AWS. The previous approach was simply to re-brand IT and System Operations as "DevOps" despite having no "Dev" experience to speak of. Adopted a more integrated, SRE-style of working alongside development teams, and (mostly) ended the practice of dev teams "tossing things over the fence" to some Ops team in the parts of the org that the Seattle SRE team supported.

Staff Software Engineer (October 2016—October 2018)

Ryan led the development of multiple tier-1 services as part of the educational content authoring pipeline, leveraging REST, GraphQL, API design, AWS, Amazon ECS, Docker, Terraform, ePubs, and security best practices. Led the technical direction of the projects, socialized them, documented them, and provided ongoing guidance around their design and use.

- Lead the development of the authoring component of McGraw Hill's SmartBook 2.0 product, and the internal system which indexes authored content, builds ePubs, and encodes images/video for McGraw Hill's ePub CDN.

- Member of the core team developing a newer approach to deploying applications, which leveraged continuous integration and continuous delivery. While many applications and processes were built around larger deployments occurring every few weeks, this team was charged with developing and dog-fooding newer processes which allowed deployments that were both more frequent and more reliable.

- Introduced a more hands-on monitoring style, which enables development teams to be more actively engaged in their own operations instead of relying exclusively on an external, third-party vendor used by other groups in the company. This enabled us to provide significantly-lower MTTR during incidents, and by digging into application-level metrics (instead of exclusively infrastructure-level metrics), we were better able to provide valuable data which addressed KPIs/SLOs for the kinds of experiences our customers were having.

- A member of the core team that was migrating all new infrastructure to "Infrastructure-as-Code" tooling such as Terraform, Packer, etc. Identified patterns across applications, and began the effort to streamline infrastructure maintenance with shared, re-usable Terraform modules.

Perimeter of Wisdom, LLC

Co-Owner, CTO, Producer (February 2015—2018)

On the technical side, Ryan built the entire “The First-Time Offender’s Guide to Freedom” website, soup to nuts. Ryan also performed all of the production work on the eBook, authored by E. M. Baird.

- Leveraged modern tools to build the front-end, including Bootstrap, LESS, JavaScript, Gulp.js, npm, Bower. Ryan built the back-end in PHP 5.6, using HHVM and Nginx, MySQL, Redis, Slim Framework, Monolog, Pimple, Twig, Guzzle, Doctrine, Phinx, and Symfony components. Ryan deployed the application using Ansible, and developed the application in a Vagrant environment running Ubuntu.

- Runs the unit, integration and functional tests using PHPUnit, Behat, Mink, and Selenium. Ryan leverages Amazon SES for sending email, Amazon S3 for static file storage, Stripe for payment processing, Linode for web hosting, MaxMind IP-based geolocation, and Google Books and Dropbox for ensuring that customers always have the latest errata fixes.

WePay — Redwood City, CA

DevOps Engineer (April 2015—September 2016)

- Improved how WePay provisioned cloud infrastructure, deployed updates, managed security patches, monitored applications and infrastructure, and streamlined the process of planning, developing, deploying and maintaining new micro-services throughout the company.

- Led the cross-company effort to upgrade the monolithic application’s software stack from PHP 5.4 to PHP 5.6. This required cross-team collaboration across all of the major engineering teams, QA, and replacing over 200 servers across multiple environments with zero customer-facing downtime.

- Maintainer of multiple tier-1 systems including Artifactory, GitHub Enterprise, Toran Proxy and Phabricator.

Senior API Engineer (April 2014—April 2015)

- Developed new API endpoints to help expand WePay's business and support its partners.
- Was instrumental in designing/developing WePay's MFA-as-a-Service offering ("System and Methods for User Authentication across Multiple Domains" (US15042104; Pending)).
- Heavily involved in the security of WePay's products, coordinating fixes with teams against other priorities, and fixing the issues himself in many cases.

Amazon — Seattle, WA

Web Development Engineer II, Amazon Web Services (March 2010—April 2014)

- Adapted CloudFusion into the AWS SDK for PHP — AWS's SDK for rapidly building cloud-based web applications (launched September 2010). Invested heavily in supporting the needs of developers by taking the time to listen and understand the needs of developers, and is involved in PHP-related industry groups on behalf of AWS.
- Worked with the AWS Elastic Beanstalk team to provide PHP support for the platform (launched March 2012). In addition to working with the PHP community to determine the configuration for a PHP container that would fit the greatest number of developers, he developed a rigorous internal test suite for testing containers which has been used as the basis for testing by other language-specific teams. He also had early input on adding support for `git push` deployments.
- Heavily involved in the creation and development of the AWS SDK for PHP 2, which takes into account the numerous changes in the PHP language and community since Tarzan/CloudFusion was first written in 2005 (launched November 2012).
- Worked with the AWS Design team on the AWS Management Console, where he lends his experience as a web developer and software engineer to bridge the gap between the design and engineering disciplines in an effort to build a high-quality, robust, user-friendly console for interacting with Amazon Web Services.

CloudFusion (née Tarzan) — Open-Source Project

Creator and Developer (Early 2005—March 2010)

- CloudFusion is a fast, powerful PHP toolkit for building awesome, cloud-based web applications in a fraction of the time! Design decisions are made in the best interests of performance, ease of use, and overall usability. Goals are to provide a high-performance developer toolkit for leveraging Amazon's

cloud infrastructure, to grow the community and, and to build useful user-centric apps based on the toolkit.

- Amazon Web Services hired me to fork this project in 2010. It became the AWS SDK for PHP.

Rearden Commerce (now Deem) — Foster City, CA

Senior User Experience Developer (July 2008—March 2010)

- Supported the user experience team, Java developers, and widget development teams. This involved prototyping new features, integration of those new features, migrating JavaScript code from older frameworks to YUI, and educating other teams on the value of high-quality front-end code — all while placing a huge emphasis on writing front-end code with better performance, faster load times, and improved accessibility across the board.

WarpShare — Morgan Hill, CA

Co-Founder and Chief Information Officer (September 2006—March 2010)

WarpShare was working to bridge the gap between digital piracy and the economics of the RIAA/MPAA industry groups.

- We developed a P2P protocol that was more efficient than BitTorrent called CleerPeer ("Hive-based Peer-to-Peer Network" (US8103870B2)).
- We designed and began development on a social network focused around digital media, and "gamification" around tagging and improving content (over automated data sources).
- We attempted a business model where users could support/sponsor content by interacting with advertising that designed to be a part of the media experience, instead of interrupting your media experience (similar to Apple's iAd platform, which came later). We likened this to people who looked forward to seeing the latest "I'm a Mac" ad that was popular at the time.
- By interacting/engaging with the content-targeted advertising, advertisers would *sponsor* the download, paying the 99¢ per song that we would hold in escrow (this was similar to, but not the same as, the failed business model for Readability, which came later). One of the key differences between WarpShare and Readability's business model was that although both services were designed to collect money on behalf of the copyright owner, Readability intended to keep any forfeiture due to the lack of a deal, WarpShare's plan (not vetted by a lawyer) was to give the money away to charity.
- Failed because: team was too small; team lacked the required expertise in advertising; funding dried up as the US entered the *credit crisis* from 2007–2009; tried to do too much up-front; early mistakes spending money on *starting a company* instead of developing a consumer product.

SimplePie — Open-Source Project

Creator and Co-Developer (July 2004—October 2009), contributor (ongoing)

- Ryan is the creator, evangelist, and co-developer of the SimplePie project — a PHP library that enables web developers to simply and easily integrate news feeds into their websites and web applications.
- After recruiting additional development resources in June 2005, Ryan began to shift from a primarily development-focused role to a primarily people-focused role, where he currently works to ensure that people are aware of, and can easily use SimplePie through support, documentation, tutorials, plugins, and evangelism.
- SimplePie was integrated into WordPress, Drupal, MODx, and several other large projects written in PHP. If you've ever used WordPress since 2006, you've used SimplePie with or without knowing it.

Self-Employed

Consulting and development services (2007—2009)

- As a freelance developer, Ryan leverages a deep understanding of best practices in front-end development, layout and design, information architecture, usability, accessibility, and web culture to provide value to clients. He provides guidance to people and teams about how to maintain best practices after the project ends.
- Took on various gigs to stay afloat when WarpShare was broke during the credit crisis.

Yahoo! — Sunnyvale, CA

Front-end Developer (Contract), Yahoo! Messenger (November 2007—January 2008)

- Ryan lead the front-end development of the Spring 2008 re-launch of the Yahoo! Messenger website. He collaborated with a core team of developers to provide increased usability, accessibility, organic search engine optimization (SEO), and simplified maintenance, resulting in exceptionally tuned performance for 29 locales.
- Ryan was involved in tuning the front-end stack for performance, where they employed semantically valid HTML/CSS, caching, gzipping, image spriting, code minification, and reduced HTTP requests, resulting in exceptional performance.

Stryker — San Jose, CA

User Interface Developer (May 2005—September 2006)

- Ryan was a core member of the team tasked with re-building the company intranet site around Oracle Portal. His time was spent writing and discussing functional and technical documentation, conducting usability interviews, and creating a fresh UI that employed user-centered design principles, web standards, and fancy new AJAX tech.

- Ryan was also a member of the Endora Marketing Team, which was geared towards spreading information about the company's move to Oracle's ERP software. In that capacity, Ryan maintained the Endora website, wrote numerous articles for the monthly newsletter, interviewed project leads, and created fun little ERP-related polls to help drive interest in the project.

- Ryan worked with the eBusiness team to improve maintenance and development for the UI of the GlobalSource project. He also re-engineered the Stryker Endoscopy public site to follow modern web standards, and built a PHP-based templating system for the site that significantly sped up development.

Digital Impact — San Mateo, CA

Production Specialist (March 2004—April 2005)

- Ryan coordinated with Campaign Managers on email campaign integration, with responsibility for email content and change requests, and ensuring that the content format was consistent with client requirements. He performed the quality tracking and reporting of campaign integration-related metrics, and consulted and troubleshot on text and HTML templates.

- Ryan maintained HTML code guidelines, provided optimal design and processing, and provided suggestions for strategic and process improvements. He also acted as syndication expert for the internal RSS development team.

- Ryan's client experience included Banana Republic, SBC (now AT&T), Hewlett Packard (HP), Sony Style, Lexus, MAC Make-up.

Truncated

Earlier experience from before I graduated college is available upon request.

Recommendations

A full list of recommendations can be found on my LinkedIn profile. Here are a few of my favorites.

Will Curran

Head of Developer Metrics and Insights, Google Cloud Platform

> “Ryan is one of the most customer focused individuals I have worked with. He takes great pride in his work and is constantly evaluating how to improve the end user experience. He backs his opinions with customer feedback and data, and I often relied on Ryan to help me deliver a better experience to user, in a short period of time.”

Brendan Dixon

Software Development Manager, formerly Amazon Web Services, Microsoft

> “What I appreciate about Ryan is his obsession to detail and customers. Ryan refuses to let business politics to ever interfere with doing what is best for customers. He invests himself to discover the best solutions and then make them available. I wholly trust Ryan's evaluation of front-end engineers and Information Architecture. Ryan would make a solid contribution to any team requiring solid front-end skills blended with a deep customer concern.”

Brian Thompson

Business Intelligence Development Lead

> “Ryan has sort of become my informal mentor regarding my web development role within Amazon. He's passionate about what he does, he's extremely talented and his "can-do" approach to projects makes him valuable on any team he becomes a part of. Perhaps even more importantly than his direct contributions to a given role however is his steady presence in stress, the ability to absorb (and apply) new information and technology quickly and and unquestioned desire to see those around him succeed. I pride myself in my profession and look up to Ryan as a mentor, a colleague and a friend. Ryan Parman is an outstanding, well-rounded and positive leader who inspires confidence in those who appeal to him for technical help or simply solid advice. I hope Amazon never loses him for greener pastures.”

Chuck Mortimore

Head of Security Products at Visa

> “Ryan gets it done. Usually you don't even have to ask... it just gets done.”

Adrien Cahen

Full-stack Javascript Engineer, Airbnb, formerly Yahoo!, Twitter

> “Ryan is a rock star. Through his work on SimplePie, he has a healthy understanding of PHP and server-side concerns. He is extremely proficient in all aspects of modern web development [...]. He is aware and respectful of standards-body recommendations, but he knows that in the end, user satisfaction (as opposed to developer comfort) is most important. [...] [Ryan managed] to go above and beyond the call of duty by proposing and implementing creative solutions to the hurdles that appeared along the way.”

Brian Emmett

Software Engineering Manager, Google, formerly Apple, Netflix

> “What has always impressed me about Ryan was his internal motivation for continual improvement. Whether it's creating software in his spare time or researching and implementing bleeding-edge UI techniques, I've always admired his drive. Coupled with a rich technical acumen and superior interpersonal skills, it was always a pleasure to work with him [...].”

Matthew Clower

Chief Software Architect, WePay

> "Ryan has both an excellent technical perspective and the drive to fight for the common user. He has a very wide understanding of development's, web services', and online communities' concepts and finds the best way to accomplish the tasks at hand. The caliber of his work is a rarity among his field and he pulls knowledge and services from the most applicable sources while interfacing quickly, effectively, and concurrently with design, development, strategic, marketing, and executive teams."

Vada Dean

Principal at Dean & Associates

> "Ryan is one of those rare people capable of tapping deep creative, technical, and operational proficiency. Capable of solving difficult problems while marshalling external/internal resources to deliver high quality results within budget and on-time. SimplePie provides a good example of Ryan's abilities. He cast the vision, recruited a partner, provided significant chunks of code, and evangelized the project across the Net."

Groups & Accomplishments

- Editor/Producer/Publisher for the book "Federal Probation Bible, 2022–2023 Edition" written by E.M. Baird. (ISBN: 9780578992693)
- Voting Representative for AWS, PHP Framework Interoperability Group (2012–2013)
- Member, RSS Advisory Board (2007—2009)
- Patent, "Hive-based Peer-to-Peer Network" (US8103870B2)
- Patent, "System and Methods for User Authentication across Multiple Domains" (US15042104; Pending)
- Student guest speaker for the 2004 Silicon Valley College graduation ceremony.

Education

Carrington College California (née Silicon Valley College) — San Jose, CA

- Bachelor of Arts, Design and Visualization, November 2003. 3.84 GPA
- Related Coursework: Web, graphic, multimedia, and publication design.