Ryan Parman

Summary

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.

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 reliably. 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.

Truncated

Earlier experience is available upon request.

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.