Matt Scilipoti — Agile Software Developer

Contact: matt@scilipoti.name | (443) 538-8656 | https://example.com

What can I do for you?

Jack-of-all-trades software developer focusing on full life-cycle, agile development, teaching, and production reliability. Experienced with Ruby on Rails, web application architecture, DevOps, and developer training.

How did I earn these skills?

My interest in programming began as a hobby over 30 years ago. Professional development, consulting, training, and conference presentations—combined with hands-on work across startups and enterprise—have produced a broad base of practical skills in web development, databases, DevOps, and agile practices. Early adoption of Agile (2002) and Ruby/Rails (2002/2005) informed my approach to coaching and building maintainable systems.

What are my skills?

- **Core Skills:** Ruby, Ruby on Rails, DevSecOps, JavaScript, Docker, Microsoft SQL Server, PostgreSQL, Architecture, CSS, EngineYard, HTML, Legacy migration, Resque, Visual Basic, Accessibility

Work Experience

Senior Software Engineer — Space Telescope Science Institute (STScI)

2016-07 — present

Senior Software Engineer and DevSecOps on the Viewspace project — a custom digital-signage and playlist platform for museums and libraries. Led full-stack development, built local playback agents and IndexedDB caching for offline-friendly playback, and improved deployments and developer experience through containerization and GitLab CI/CD, increasing reliability while lowering streaming costs.

- Built the Rails ingest and playlist API, shipped a lightweight local playback agent for Raspberry Pi, and added IndexedDB-based caching — enabling devices to play content locally rather than stream continuously.
- Migrated deployments from Capistrano to GitLab CI/CD, containerized services with Docker, and introduced VS Code devcontainers to make builds reproducible and speed developer onboarding.
- Implemented programmatic Vimeo tooling (uploads, metadata, environment scoping) and then a hybrid caching model (automatic background sync + integrity checks) that substantially reduced streaming bandwidth and monthly hosting costs while preserving 24/7 local playback.
- Added monitoring, watchdogs, and recovery logic in playback clients to detect and recover from stalled playlists; rolled out synthetic checks to alert operations and reduce mean time to detect issues.
- Result: replaced an unreliable commercial product, scaled to 300+ venues, shortened update cycles from months to immediate web deployments, and materially improved playback reliability and cost-efficiency (see Phase 4–6 notes for metrics).
- Company: Space Telescope Science Institute (STScI)
- Project name: Viewspace
- Position: Senior Software Engineer (listed in resume)
- Start date: 2016-07
- End date: (unknown) — please provide if available
- Team size / role: (unknown) — please provide whether you were sole engineer, lead, or part of a team
- End date: still at STScI (present)
- Team size / role: Two senior developers on the project (you + 1 other senior dev)
- Work arrangement: Hybrid until 2019; Remote since 2019
- During Phase 1 we also learned and supported the aging commercial digital-signage tool in production while planning and building the replacement, which helped with migration planning and informed feature priorities.
- Video upload and management UI
- Playlist creation and scheduling

- Venue/device selection and remote configuration
- Local playback agents for offline/low-bandwidth playback (devices played video locally rather than streaming continuously)
- On-prem video storage provided by ITSd (videos stored on STScI infrastructure)
- Deployment using Capistrano to on-prem servers
- DevOps responsibilities: deployments, rollbacks, monitoring, incident response
- Microservice architecture mentioned in original resume (details to confirm)
- Production capacity limited to ~300 venues due to bandwidth constraints. The initial requested cap by management was 100; we later supported up to 300 for Phase 1. This cap was established so ITSd could provide concierge-level service to each venue.
- On-premises storage and bandwidth limits influenced architecture decisions (local playback agents, prefetching, etc.)
- Collaborated with stakeholders at STScI (content teams, ITSd for storage, and venue partners)
- Worked with an external consulting firm/contractor who had previously built a partial web app; part of the work was re-scoping and replacing/refocusing that work
- Replaced an unreliable, paid commercial signage tool with a free, reliable Viewspace web deployment.
- Increased viewers per venue from ~100 to ~200–400 (roughly a 100–300 increase depending on venue), and provided immediate updates to venues via web deployments rather than long local install/update cycles.
- Prior tool often required local installs and could take 6+ months for venues to update; Viewspace delivered immediate updates and could run on low-power hardware (e.g., Raspberry Pi), lowering requirements and deployment friction.
- Standardized deployment and rollback procedures; reduced playback failures and improved availability (quantitative metrics can be added if available).
- Backend: Ruby on Rails web application
- Frontend: Rails (server-rendered) and some light JavaScript
- Local playback agents: lightweight clients capable of running on Raspberry Pi and similar low-power devices
- Deployment: Capistrano to on-prem servers
- Storage: On-prem ITSd storage for video assets
- DevOps: on-prem deployments (single server). ITSD managed postgres, monitoring, and incident response
- "Led development of Viewspace, replacing an unsupported commercial digital-signage solution by building a custom video playlist platform tailored for museums and libraries."
- "Implemented local playback agents and server APIs to enable offline-friendly, 24/7 playback across up to ~300 venues."
- "Managed on-premises deployments with Capistrano and on-prem video storage; standardized release and rollback procedures."
- "Collaborated with stakeholders and an external contractor to re-scope requirements and focus engineering effort on high-impact features."
- Provide answers to the open questions above.
- I'll use the confirmed details to produce 2–3 resume bullet variants (technical and leadership), a 1–2 sentence summary, and a LinkedIn-ready project blurb.
- Optionally: add screenshots, architecture diagram, or short code snippets (if repository or artifacts exist) to strengthen the write-up.
- Ramp up content production and improve viewer engagement by making playlists more dynamic and shortening continuous play length while ensuring all videos still appear over multiple playthroughs.
- Informal Educators and subject-matter experts indicated these changes would improve informal learning and visitor engagement; we incorporated their guidance into playlist rules and templates.
- Collaborated with Informal Learning SMEs to design playlist randomization strategies that kept viewers interested while guaranteeing coverage of all videos after multiple cycles.
- Converted static playlists into a templated system: playlists could now contain videos and/or nested playlists, enabling reusable composition.
- Implemented complex playlist templates that allowed conditional/random selection of videos by type at specified points in a playlist (for example: pick 1 random 'short' video, then 2 from 'exhibit' category, etc.).
- Playlist templates were modeled on the server side and rendered into play sequences for local playback clients; logic balanced randomness with coverage guarantees.
- Work included API endpoints for template creation, previewing play sequences, and exporting playlists for devices.
- Shortened average continuous play time per venue while maintaining content coverage, improving viewer retention.
- Enabled richer, varied visitor experiences with more dynamic playlists and targeted content sequencing.
- Informal Educators reported improved informal learning and engagement after rollout, validating the SME-driven playlist rules and templates.

- Move the project from ad-hoc on-prem deployments to a reproducible, container-based CI/CD process and improve developer environment parity and onboarding.
- Migrated source control and CI to GitLab, creating pipeline definitions to build, test, and package the Rails app and playback agent into Docker containers.
- Reworked on-prem deployment workflows to pull images from an internal registry and run containers on target servers, replacing Capistrano-based releases for most components.
- Introduced VS Code devcontainer configurations so developers could run a consistent development environment (matching runtime dependencies and Docker-based services) locally and in CI.
- Reproducible builds and faster, automated deployments via GitLab CI/CD and Docker reduced deployment friction and improved rollback reliability.
- Developer onboarding and environment parity improved through devcontainers, reducing "works on my machine" issues and setup time.
- Migrated deployments from Capistrano to GitLab CI/CD and Docker containers, standardizing build pipelines and on-prem container deployments.
- Implemented VS Code devcontainer configurations to provide consistent developer environments and streamline onboarding.
- Designed and implemented a templated playlist system allowing nested playlists and conditional/random selection, improving viewer engagement during the 8-month content ramp-up.
- Partnered closely with Informal Learning SMEs to translate educational goals into playlist templates and randomization rules that balanced novelty with content coverage.
- Remove on-prem bandwidth constraints and enable larger-scale growth by offloading video hosting to a third-party CDN/service.
- Migrated video hosting to Vimeo to take advantage of unlimited bandwidth for a fixed cost, which removed the ~300-venue cap and allowed scaling to significantly more sites.
- Implemented programmatic management of Vimeo-hosted videos using their API (despite limited documentation), including uploading, metadata, access control, and environment separation (dev/test/staging/demo/production buckets/accounts/projects).
- Integrated Vimeo's embedded player into the local playback clients and web UIs, while keeping configuration to allow fallbacks for low-bandwidth or offline modes.
- Built server-side tooling to manage Vimeo resources: automated uploads, tagging, and environment scoping so each deployment environment used separate Vimeo collections/accounts.
- Worked around Vimeo API gaps by implementing retry/backoff, idempotent uploads, and audit logging of API operations; created admin tooling for video lifecycle management.
- Ensured embedded player configuration supported autoplay/kiosk modes and same-origin considerations for secure embeds in venue displays.
- Removed bandwidth ceilings by using Vimeo's hosting, enabling growth beyond the prior limit and simplifying media delivery.
- Reduced on-prem storage and bandwidth costs while increasing reliability and playback quality across venues.
- Led migration of video hosting to Vimeo to remove bandwidth constraints and enable large-scale growth; implemented server-side tooling to manage uploads, metadata, and environment separation programmatically.
- Integrated Vimeo embedded player into Viewspace clients and built resilient API workflows (retry, idempotency, audit) to compensate for sparse documentation.
- Major browser vendors introduced autoplay/sound policies that require user interaction for videos with sound; behavior differs across browsers and versions, causing previously working embeds to stop autoplaying reliably.
- Vimeo's hosting and embedded player introduced unexpected operational challenges: increased bandwidth billing (now > $10k/month), intermittent player failures, playlists stopping mid-playlist, and quality/bandwidth fluctuations across venues.
- Browsers: inconsistent autoplay policies and limited, evolving documentation; different config flags and allowed behaviors for muted autoplay, cross-origin embeds, and kiosk/autoplay modes.
- Vimeo: we used APIs and embedded players in ways not anticipated by their best practices; documentation and support were limited, leading to brittle integrations.
- Increased operational costs (bandwidth billing > $10k/month) and reduced reliability for venue playback.
- Random playback failures and mid-playlist stops created poor visitor experiences; debugging was difficult due to differing browser behavior and sparse Vimeo error telemetry.
- Implemented browser-specific embed configurations and heuristics to detect autoplay policies and fall back to muted autoplay or user prompts when necessary.
- Added retries and watchdogs to playback clients to detect stalled playlists and attempt restart or skip to next item.
- Rolled out monitoring and synthetic checks to catch failing players and alert operations.
- Added a quick fix using Vimeo's looping capability: produced several long, looping videos that simulated templated playlists (so venues could play a single looping file rather than run complex JS), which reduced client complexity and improved reliability in many cases.

- Evaluate alternative hosting/CDN options with predictable pricing and server-side player control (self-hosted CDN, cloud object storage + CDN, or vendor with stronger API/support SLA).
- Implement a hybrid playback model: prefer embed/CDN streaming when reliable, but prefetch and locally cache critical assets for offline/resilient playback in constrained venues.
- Advocate for staged browser testing matrix (Chrome/Firefox/Safari versions) and an adaptation layer in the playback client to unify autoplay behavior.
- Work with Vimeo (or vendor) to obtain clearer SLAs or engineering support for embed-related issues; consider contracts that include support for production-scale use.
- Diagnosed cross-browser autoplay regressions and implemented client-side fallbacks, web heuristics, and monitoring to improve playback resilience.
- Investigated Vimeo integration and operational costs (> $10k/mo), designed hybrid caching and CDN strategies to reduce bandwidth costs and increase reliability.
- Implemented a pragmatic looping-video mitigation using Vimeo's loop feature to simulate templates and reduce client-side JS complexity, improving playback reliability for constrained venues.
- Reduce operational bandwidth costs and increase playback reliability by caching videos locally on devices while preserving contractual obligations to refresh content regularly.
- Re-architected playback flow to programmatically download the highest-quality version of each Vimeo-hosted video and store it in the browser's IndexedDB for local playback.
- Switched from forced reliance on the Vimeo embedded player to a local player stack, increasing accessibility and control over playback features and behavior.
- Implemented a 24-hour periodic sync policy (required by contract) so devices must connect at least once per 24 hours to fetch new or updated videos.
- Developed background sync and retry mechanisms to download and update video assets when network conditions permit; implemented integrity checks and versioning for cached assets.
- Built admin tooling to map Vimeo video IDs to local cache keys and to schedule/update content for different environments (dev/staging/prod).
- Substantially reduced streaming bandwidth and monthly hosting costs while improving playback reliability (local playback from IndexedDB eliminated streaming interruptions).
- Increased accessibility and player customization options since we were no longer constrained to the Vimeo embed API and player behaviors.
- Re-architected playback to download and cache Vimeo videos in IndexedDB and play locally, reducing bandwidth costs and dramatically improving playback reliability across venues.
- Implemented background sync, integrity/version checks, and a 24-hour sync policy to comply with contractual requirements while keeping content fresh.
- Phase 6 implementation is live in beta; early results show a massive drop in streaming bandwidth usage and overwhelmingly positive feedback from beta testers praising the local-first playback experience. Precise metrics are being collected and will be added when available.
- Worked on Viewspace from day one at STScI to the present (with short breaks to support other apps). Supported over 17 applications across the Office of Public Outreach (OPO), providing development, maintenance, and operational support.
- OPO's primary mission areas include sharing the science of space telescopes via news, informal learning, conference presentations, and public installations (airports, visitor centers, etc.).
- Other apps supported include MediaLibrary (managing news articles, images, videos used across OPO publications and displays).
- Cross-browser testing: established practices and automated checks using CrossBrowserTesting.com to validate behavior across browsers and OS combinations.
- OPO-DevOps: built an internal operations dashboard/tooling to inventory OPO applications and servers, run basic tests, show status, and manage multiple containerized services.
- Accessibility: began an accessibility initiative starting with automated HTML checking using W3C tools and integrated checks into the QA workflow.
- Improved cross-browser QA and reduced regressions by formalizing testing practices.
- Increased operational visibility and reduced mean time to detect issues through `OPO-DevOps` dashboards and basic synthetic checks.
- Laid groundwork for accessibility improvements by integrating W3C HTML checks into development pipelines.
- Supported 17+ OPO applications from initial development to production, providing feature work, maintenance, and operational support across the division.
- Built `OPO-DevOps`, an internal ops dashboard to inventory apps and servers, run tests, manage containers, and surface status for operations and developers.
- Established cross-browser testing practices using CrossBrowserTesting.com, reducing browser-specific regressions and increasing test coverage.
- Initiated accessibility tooling and processes (W3C HTML checks) to improve frontend accessibility and integrate checks into QA.

Senior Instructor and Instructional Coach — General Assembly

2014-09 — 2016-07

Co-instructor and coach for the Web Development Immersive (WDI) course. Delivered curriculum, mentored instructors, and prepared students for developer roles.

- Co-delivered an intensive full-stack curriculum covering HTML/CSS, JavaScript, Ruby on Rails, Angular, and Express.
- Onboarded and coached instructor teams, providing lesson reviews, feedback, and teaching best practices to improve student outcomes.
- Mentored students through capstone projects and supported career readiness activities that helped graduates land developer positions.

Senior Developer — LearnZillion

2012-08 — 2014-09

Senior developer on a multi-tenant Ruby on Rails platform for teacher professional development.

- Built multi-tenant features using PostgreSQL and ElasticSearch to power full-text search and content personalization.
- Implemented background processing with Resque and improved reliability of asynchronous workflows.
- Automated server provisioning using Chef and drove the team's adoption of Continuous Delivery practices.
- Evaluated and piloted containerization (Docker) as a path toward reproducible deployments.

Senior Agile Developer (contract) — Federal Reserve Board of Governors

2012-01 — 2012-08

Contract work for the Board's eDiscovery and Legal units to improve FOIA processing and eDiscovery workflows.

- Extracted and indexed millions of emails across Lotus Notes stores to enable high-speed search and review.
- Implemented search-driven pipelines (Sphinx-based) and prediction models to reduce manual review and redaction time.
- Improved processing reliability for encrypted attachments and heterogeneous data sources, reducing time-to-deliver for FOIA requests.

Senior Agile Developer — Groupsite.com

2011-01 — 2012-01

Worked on legacy and migrated Rails applications and supported hosting and operations.

Senior Agile Developer — SmartLogic Solutions

2010-06 — 2011-01

Led a 4-person web application project and improved QA and architecture decisions.

- Led a cross-functional team to deliver a Rails-based web application from inception to production.
- Established acceptance testing with Cucumber, improving regression coverage and release confidence.
- Shaped UX and architectural trade-offs to improve performance and maintainability.

Senior Agile Developer — Traffipax, Inc.

2008-12 — 2010-03

Main Rails developer for traffic management and safety applications.

- Replaced a legacy Java system with a Rails-based solution, reducing turnaround from 10 days to 1 day for critical workflows.
- Automated server provisioning and deployment to speed releases and reduce errors.
- Implemented monitoring and log-based alerts to support field technicians and improve system reliability.

Senior Agile Developer — Medical Decision Logic (MDLogix)

2008-05 — 2008-12

Worked on clinical trial and patient management applications, modernizing legacy Rails apps and mentoring teams on Rails best practices.

- Modernized legacy Rails applications, improving maintainability and performance for clinical-trial workflows.
- Mentored engineering teams on Rails conventions, TDD, and agile practices to increase development velocity and code quality.
- Tackled technical debt and simplified deployment procedures to shorten release cycles.

Senior Developer — Mind Over Machines, Inc.

2000-06 — 2000-11

Clients ranged from legal service providers to small manufacturing concerns. Visual Basic desktop development: client server and n-tier architecture. Provided DBA support for MS SQL and MS Access. One project involved replication of databases across the Internet. This engagement evolved into a working relationship between my company and MOM, Inc.

Agile Software Craftsman — Possiamo Consulting LLC

2000 — 2016-07

I am an independent consultant providing custom software and training. Work includes e-learning, bioinformatics, knowledge management, rules-based engines, insurance processing, inspection validation, and distributed document management and workflow. We create web applications (Rails & ASP.NET) and Windows desktop applications. Middleware DLLs written in Ruby, VB.NET, C#, and VB6; Web Services and Remoting used as appropriate.

Senior Developer/Analyst — Ajilon

1994-07 — 2000-06

Supported industries including insurance, telecommunications, and retail. Visual Basic desktop development (client server and n-tier). Supported DBAs for MS SQL and MS Access. Migrated a large telecommunications firm from 16-bit to 32-bit applications. Introduced Object Oriented Programming practices and helped bring .NET to clients in 2000. Worked on importing external data feeds using DTS.

— Professional Summary

—

Matthew M. Scilipoti — Agile Software Craftsman

- Full life cycle, agile, object-oriented development
- Agile methodologies and coaching
- Web development: Ruby on Rails, ASP.NET, CSS, AJAX
- Microsoft technologies: C#, ASP.NET, VB.NET, VB classic
- System administration (Windows, Linux, macOS)
- Database administration (MS SQL, MySQL, Access)
- Training and presentations on Agile, TDD, DDD
- Familiarity with Open Source community and tools

Education

{{#education}}{{/education}}