Sprocket Platform Restoration

Town Hall Presentation

Nigel Thornbrake, Head of Development

What Happened?

Sprocket went down in mid-2025 and stayed down for months.

Your gaming platform was completely offline.

Why did it take so long to fix?

That's what we're here to talk about.

The Situation: What We Lost

- Complete platform outage (web, API, Discord bot)
- All league play and scrim infrastructure
- Player stats and rankings
- 22 different services offline

The Reality: We had no infrastructure team, no backup plans, and no clear ownership of critical systems.

The Response: 8-Week Recovery

56 days of concentrated rebuilding

20+ major infrastructure commits

5 critical technical breakthroughs

Hundreds of hours of volunteer time

Cost: Zero dollars - but revealed our first problem: no budget, no process, no team structure.

The Challenge: A 22-Service System

This isn't a simple website. It's a distributed system with:

Infrastructure Layer: Traefik, Vault, Docker Proxy

Data Services: PostgreSQL, Redis, RabbitMQ, InfluxDB, Grafana, Neo4j, N8n, Gatus, Loki, Telegraf

Applications: Web UI, API, Discord bot, Image generation, 6+ microservices

Why So Complex?

This isn't overengineering - this is what modern gaming platforms require.

You need databases, caching, message queues, monitoring, logging, metrics - each piece exists for a reason.

Remove any one, and features break.

Challenge #1: Vault Unsealing

Problem: Vault required manual unsealing after every restart

The Battle: 5 different approaches over 3 days

Solution: Custom auto-initialization script with local bind mount

Challenge #2: Database Reliability

Problem: Self-hosted PostgreSQL with no backups, no HA

Decision: Migrate to managed database service

Result: Automated backups, point-in-time recovery, professional reliability

Challenge #3: Storage Migration

Problem: Self-hosted MinIO was resource-intensive and unreliable

Migration: 8-day phased move to AWS S3

Benefit: 99.999999999% durability, one less service to manage

Challenge #4: Multi-Environment Routing

Platform needed to work 4 different ways:

1. Local development (localhost)
2. LAN access (direct IP)
3. VPN access (Tailscale)
4. Public internet (real domain)

Different routing rules, certificates, DNS for each.

Challenge #5: Secret Management

Problem: Secrets scattered across Doppler, Vault, Pulumi, Docker

Solution: Hierarchical management:

- Doppler = Source of Truth
- Vault = Runtime Distribution
- Docker = Container Mounting
- Pulumi = Infrastructure Secrets

Recovery Timeline

Week 1: Foundation rebuild - Layer 1 working again

Week 2: Vault struggles - 5 attempts, breakthrough on Sept 19

Weeks 3-4: Storage migration - MinIO to AWS S3

Week 5: Platform resurrection - "Sprocket is alive!"

Week 6: Routing hell - multi-environment problems solved

Weeks 7-8: Final push - managed database, HTTPS certificates

November 8: PRODUCTION COMPLETE

What We Built

3-Layer Architecture:

- Infrastructure: 3 services
- Data Services: 9 services
- Applications: 10 services

External Services: Digital Ocean PostgreSQL, AWS S3, Doppler, Let's Encrypt, GitHub OAuth

Modern DevOps: Infrastructure as Code, automated secrets, monitoring, HTTPS

By the Numbers

Scale: 22 services, 5 networks, 15+ volumes, 20+ secret paths

Complexity: 3 Pulumi stacks, 50+ config files, 4 routing patterns

Effort: 56 days, 20+ commits, 5 breakthroughs, countless hours

The Real Problem: Organizational Failure

This wasn't just technical - it was organizational:

1. No Infrastructure Team
2. No Knowledge Transfer
3. No Backup Plans
4. No Budget Allocation
5. No Succession Planning

We got lucky. One person had the skills. Next time, we might not be so fortunate.

Why So Hard? The Honest Truth

Distributed Systems: 22 services, complex networking, each with quirks

Security: Multiple OAuth providers, Vault policies, certificates

Multi-Environment: Different routing, certificates, DNS for each pattern

DevOps: Complex orchestration, no simple restart solutions

The Over-Engineering Reality

Designed for: 100k+ users across dozens of organizations

Reality: Much smaller scale, single organization

We paid dearly: 22 services instead of 5-6, complex multi-tenancy, specialist knowledge required, longer development cycles, higher barriers to entry

New Direction: Simplified by Design

Old: One master deployment for all organizations

- Massive centralized infrastructure
- Complex multi-tenancy
- Scale that never materialized

New: One deployment per organization

- Tailored infrastructure
- Simple, focused architecture
- Independent scaling

Sprocket v2: The Future

Infrastructure: 22+ services → ~6 core services

- Single-organization focus
- Standard deployment patterns
- Reduced orchestration dependency

Open Source: ELO and matchmaking systems opening up

- Detailed Proposal: Sprocket v2 Unified Matchmaking
- Community ownership and audits
- Simplified integration

What We're Doing Now

Better Documentation: Architecture guides, deployment steps, troubleshooting

Managed Services: Cloud PostgreSQL, AWS S3, external expertise

Automation: Vault unsealing, secret provisioning, health checks

Reduction: Removed MinIO, consolidated config, simplified routing

But: Some complexity unavoidable. That's why we're building v2.

This Cannot Happen Again

Relying on emergency volunteers is not sustainable.

What if:

- Crisis hits during busy period?
- Experts aren't available?
- Nobody knows the solution?
- Someone burns out?

Answer: Everything falls apart again, possibly for good.

We Need Your Help

Transition from crisis response to sustainable operations.

We need people who can:

- Maintain infrastructure
- Debug production issues
- Improve documentation
- Assist with deployments
- Learn alongside us

We Need Your Help (Cont.)

You don't need to be an expert. You need:

- Willingness to learn
- Time to commit
- Interest in the community
- Not afraid of challenges

How You Can Help

Infrastructure & DevOps: Learn Pulumi, help deploy, improve monitoring

Documentation: Write guides, create tutorials, troubleshoot

Testing & QA: Test deployments, verify health, report issues

On-Call Support: Be available for incidents, debug urgent issues

Small Stuff: Review PRs, update docs, improve scripts, test locally

What You'll Learn

Marketable skills companies pay six figures for:

- Infrastructure as Code (Pulumi/Terraform)
- Container Orchestration (Docker/Kubernetes)
- Secrets Management (Vault, Doppler)
- Database Administration (PostgreSQL, Redis, Neo4j)
- Monitoring (Grafana, Loki, InfluxDB)
- DevOps Best Practices
- Production System Debugging

What Happens Next

Short Term (3 months):

1. Stabilize production
2. Set up monitoring
3. Create on-call rotation
4. Document solutions

What Happens Next

Medium Term (6 months):

1. Build infrastructure team
2. Train new volunteers
3. Improve automation
4. Reduce single dependencies

What Happens Next

Long Term (1 year):

1. Evaluate managed alternatives
2. Build sustainable operations team
3. Eliminate single points of failure

How to Get Involved

Contact Us:

- Talk to us after this session
- Join #infrastructure on Discord
- Attend weekly meetings
- Review docs, ask questions

How to Get Involved

What We Want:

- Curiosity and willingness to learn
- Time commitment (hours/month helps)
- Interest in DevOps/infrastructure
- Team players who communicate

How to Get Involved

What You Get:

- Production system experience
- Mentorship from engineers
- Resume-worthy projects
- Keep community alive

Q&A: Common Questions

Q: Why so complicated?
A: Modern gaming platforms need this architecture. We serve thousands of users.

Q: Why so long to fix?
A: No team, no process, no budget. Had to start from scratch.

Q: What if experts leave?
A: That's why we're building a team. No single person should be critical.

Q&A: Common Questions

Q: Time commitment?
A: Flexible. 2-4 hours/month helps. Something > nothing.

Q: I don't know Docker/Kubernetes/Pulumi, can I still help?
A: Yes! We need docs, testing, QA. We'll teach you technical stuff.

The Bottom Line

1. **Your platform was dead.** We recovered it through emergency volunteer effort that revealed serious gaps.
2. **This system is complex** because modern gaming platforms ARE complex. We're simplifying where we can.
3. **We need sustainable processes.** Volunteers are essential, but we need structure, not heroics.
4. **This is an opportunity** to learn valuable skills while helping your community build something lasting.
5. **Without systemic changes**, we risk repeating this cycle when the next crisis hits.

Thank You

To the community: Thank you for your patience during the outage.

To contributors: Your effort bought us time to build something better.

To future team members: You're helping ensure this never happens again.

Let's build sustainable infrastructure together.

Open Discussion

This is your platform. You deserve to understand:

- What happened
- Why we weren't prepared
- What we're doing to fix systemic issues
- How you can be part of the solution

The floor is yours.

Ask anything. Challenge assumptions. Voice concerns. Let's talk.

Resources

Documentation: MLE Knowledge Base

- Architecture guide
- Deployment guide
- Operations runbook
- Troubleshooting guide
- Full postmortem

Resources

Code: Infrastructure as Code, Sprocket Platform

Discord: #dev-talk in Red Server

Getting Started: Read docs → Join server → Attend meeting → Pick task → Ask questions

Thank You

Let's Keep This Community Alive - Together

Questions? Comments? Want to volunteer?

Talk to us after this session or reach out on Discord.

Your platform is back. Let's build the team to keep it that way.