Leadership & Team Building

Building & Scaling Engineering Team During Cloud Migration

S: Bankbazaar decided to migrate from our datacenter to AWS, a major shift for the company. I was the solutions architect responsible for the infrastructure side.

T: Lead the migration while ensuring minimal disruption, and strengthen the teams to handle the new environment.

A: Expanded the DevOps team with 2 junior engineers and production support with 5 members, hiring for learning agility. Trained them on Kubernetes and AWS using internal sessions, hands-on mentoring, and AWS-led courses. Coordinated closely with the dev lead to split responsibilities - I owned infra readiness, he owned application readiness. We migrated in phases, starting with low-risk leaf services to uncover issues early, including unexpected DB latency over Direct Connect, which led us to reprioritize.

R: Critical services cut over with zero issues, uptime improved to 99.9%, latency dropped 50%, and cost savings met projections.

Hiring & Developing Juniors in DevOps/PS

S: During the cloud migration, we needed more hands on infrastructure and production support.

T: Hire and grow talent who could handle Kubernetes and AWS despite limited prior experience.

A: Focused interviews on learning aptitude and logical problem solving. Once hired, provided targeted training on IaC, Kubernetes, and AWS. Integrated them into live migration workstreams so they could learn in real scenarios.

R: Within months, the hires were independently managing infra tasks and supporting production, reducing load on senior staff.

Automation & Process Simplification

Service Onboarding Automation (From 1 Week to 1 Hour)

S: Post-migration, onboarding a new service to production took a week due to multiple handoffs between Dev, Infra, Infosec, and Network.

T: Remove friction and drastically shorten the onboarding timeline.

A: Built a platform abstraction layer to automate AWS/Kubernetes provisioning with secure defaults, standardized observability, and embedded Infosec scans. Negotiated a compromise on CVEs - allow deployment but run quarterly remediation drives. Changed the process so developers committed a manifest file and deployment happened on merge.

R: Onboarding dropped to under 1 hour, with improved developer satisfaction and stronger, more consistent security.

Terraforming Production Resources for Long-Term Maintainability

S: Existing production resources were manually provisioned, making maintenance error-prone.

T: Bring them under Terraform for consistency and long-term maintainability.

A: Chose to keep Terraform code extremely simple and readable so anyone in the org could contribute. Accepted a slower migration process to ensure clarity and maintainability over time.

R: Reduced operational risk, improved onboarding for new engineers, and made production infrastructure easier to maintain.

Observability & Data-Driven Decisions

Business Insights via Observability Stack

S: Developers were debugging business/partner issues via Kafka metrics, pulling them off feature work.

T: Enable business teams to self-serve real-time insights without dev involvement.

A: Routed select Kafka streams to Elasticsearch and built Kibana dashboards with role-based access. Trained Production Support to onboard business teams.

R: Call center monitored campaigns in real time, partner issues were flagged proactively, and the system became critical enough to be separated from the core observability stack for stability.

Database Performance Diagnosis via Disk Queue Depth

S: A database was showing unexplained performance drops, and no one could pinpoint why.

T: Identify the root cause to decide on scaling.

A: Reviewed CloudWatch metrics and focused on disk queue depth, which indicated I/O bottlenecks. Used this data to justify an upgrade.

R: Performance improved, and disk queue depth became a core performance metric in our monitoring.

Security & Compliance

Security Automation & CVE Dashboards

S: Needed ISO, PCI-DSS compliance and to satisfy partner bank audits with zero critical/high CVEs.

T: Achieve security targets without slowing development.

A: Built a cross-account AWS Security Hub auto-resolution tool, created real-time CVE dashboards for developers, and ran quarterly joint remediation drives. Negotiated exceptions for unpatchable CVEs.

R: Maintained >90% Security Hub score and zero critical/high CVEs while preserving developer velocity.

Infosec Disagreement Over Cloud-Native Networking

S: Infosec wanted a traditional single-point firewall; I proposed cloud-native VPC/subnet architecture.

T: Defend the new approach while maintaining security trust.

A: Compared pros/cons, showed AWS-native controls (Istio, NLB, security groups) offered more granular security. AWS architects backed the design. CTO signed a waiver and I committed to deny-all rules and zero CVEs.

R: Adopted cloud-native architecture, preserved agility, and maintained a strong relationship with Infosec.

Distroless Image Adoption

S: Teams built and maintained base images, which was a security and maintenance burden.

T: Move to distroless images org-wide.

A: Presented the security advantages and maintenance simplicity of distroless versus custom images. Highlighted that we're not security experts and should leverage hardened vendor-provided images.

R: Organization adopted distroless images, reducing maintenance overhead and improving security posture.

Secure Secret Encryption Pipeline

S: Developers needed a way to encrypt text for git storage without exposing encryption keys.

T: Provide a secure, simple workflow.

A: Built a pipeline that took a submitted secret, encrypted it, and returned it - keys never left the system.

R: Workflow security improved, and encryption became seamless for developers.

Cost Optimization & Architecture

FinOps Cloud Cost Reduction

S: Finance requested significant AWS cost reduction to <$100K/month.

T: Identify and implement savings without hurting performance.

A: Analyzed costs, tackled major drivers (Keyspaces, S3, Kafka, RDS), negotiated data retention with business teams, improved RI coverage, implemented t-shirt sizing, and built cost visibility dashboards via CUR and Quicksight. Redesigned services for cost efficiency.

R: Costs dropped below target and governance processes were put in place to maintain control.

Email/SMS Service Redesign for Cost Savings

S: Keyspaces cost for storing email/SMS payloads was high.

T: Reduce cost without hurting performance.

A: Moved payload storage to S3, leveraging slower downstream send latency to mask retrieval time.

R: Cut Keyspaces cost significantly with no noticeable performance impact.

Cloud Migration & DR

Datacenter to AWS Migration

S: Bankbazaar migrating from datacenter to AWS; I owned infra design and execution.

T: Ensure smooth migration with minimal disruption.

A: Coordinated phased cutover starting with low-risk services, hired and trained team, handled DB latency issue by reprioritizing DB move, split responsibilities clearly with dev lead.

R: Zero issues on critical cutovers, 99.9% uptime, 50% latency reduction.

Staging Kubernetes Rebuild Failure & DR Lesson

S: Staging Kubernetes rebuild failed recently.

T: Learn from the failure to improve readiness.

A: Treated it as a DR drill, identified gaps in staging processes, and mirrored fixes to production readiness plans.

R: Improved DR readiness by applying learnings from staging.

AI / Innovation

AI Coding Tool Adoption

S: New AI coding tools were emerging with claims of higher productivity; earlier tools lacked large-codebase understanding.

T: Evaluate and recommend tools for org use.

A: Tested Cline, Claude, Cursor, and Copilot across DevOps, Java, ReactJS; ran pilot with developers; started adoption top-down with architects; enforced normal code reviews.

R: Selected Cline initially, greatly improved code documentation; later re-evaluated and moved to Claude Code.

Ambiguity & Crisis Management

COVID Secure WFH Rollout

S: COVID lockdown required 1,000 CS agents to work from home securely.

T: Deliver secure, reliable remote access under unclear conditions.

A: Implemented two-way SSL + TOTP authentication via Akamai Zero Trust, preconfigured and locked down OS, physically delivered machines, set up tiered WhatsApp support groups, solved connectivity issues with network cards/data packs.

R: Agents worked securely from home without disruption; lessons learned still applied in partial WFH setup.

Kubernetes P0 Outage During Migration

S: Datacenter Kubernetes cluster failed during migration, taking down services.

T: Restore services quickly.

A: Decided to skip repair and migrate affected services to cloud; provisioned resources, had devs prep builds, coordinated via live Teams call with all stakeholders.

R: Restored services in 4 hours, no business-day impact, management impressed with DR capability.

Annual Kubernetes Rebuild Decision-Making

S: Annual rebuild/upgrade as DR readiness sometimes triggers unexpected issues.

T: Make go/no-go calls under pressure.

A: If the fix was within the platform scope, I decided; if it was application-specific, delegated decision-making to the respective team lead with an ETA request; kept clear communications on impact and decisions.

R: Completed most upgrades successfully, aborted gracefully when needed, built trust in my judgment.

Scope Creep Pushback in Production Fixes

S: Startups often want to push fixes quickly to production.

T: Ensure changes are properly validated in staging.

A: Took hard calls to delay production deployment until adequate staging time was given.

R: Reduced production incidents from untested fixes.

Cross-Team Conflict

CVE Deployment Block — Infosec vs Dev

S: Infosec wanted to block deployments with critical CVEs; dev teams opposed.

T: Find a compromise that maintained security without blocking delivery.

A: Negotiated quarterly CVE reduction drives instead of blocking, aligned with dev cycles, maintained visibility via dashboards.

R: Security and dev teams aligned, compliance targets met without hurting velocity.