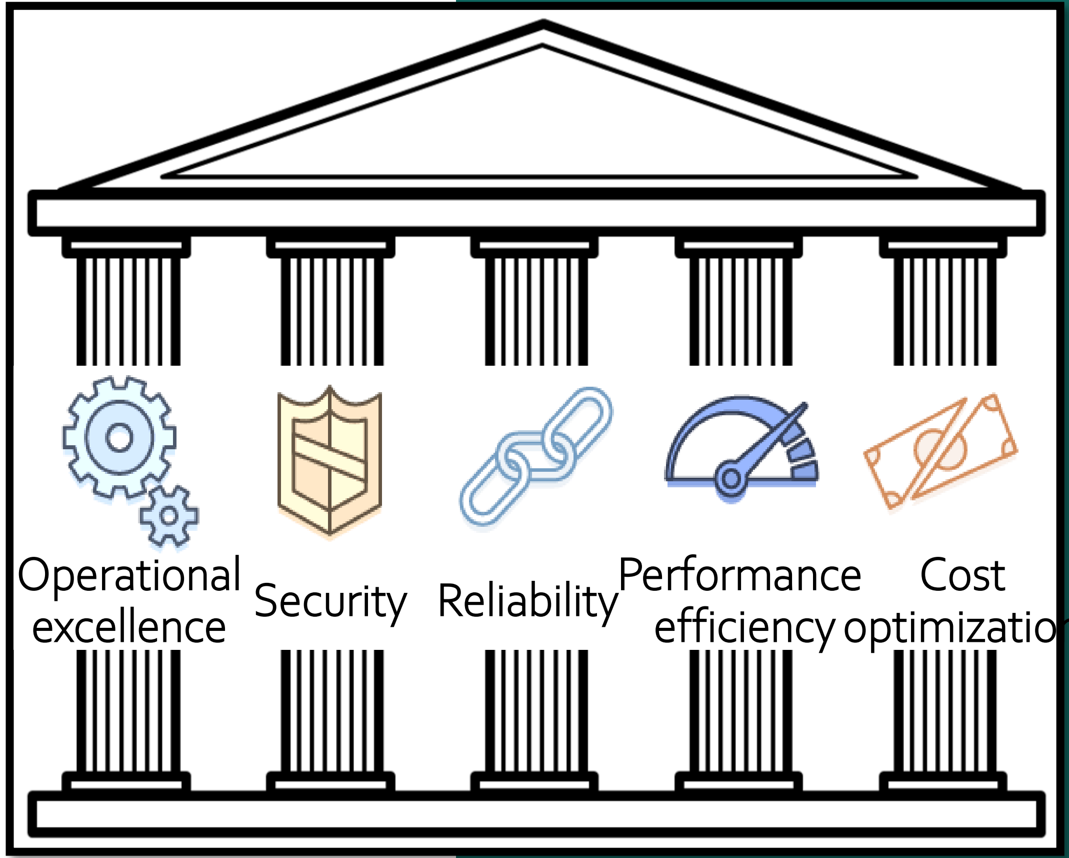
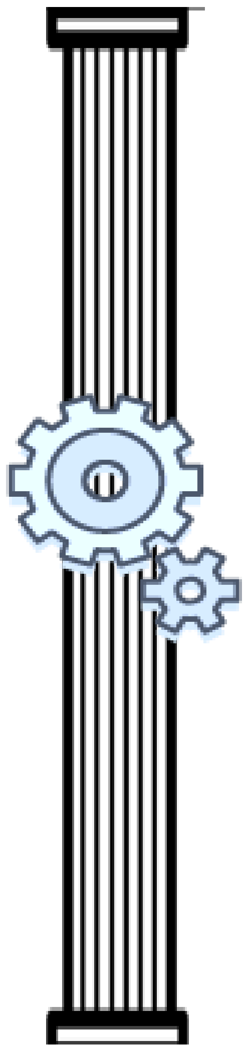
AnyCompany Architecture

Safa
Ruaa
Salsabeel
Yasmeen
30 April 2025

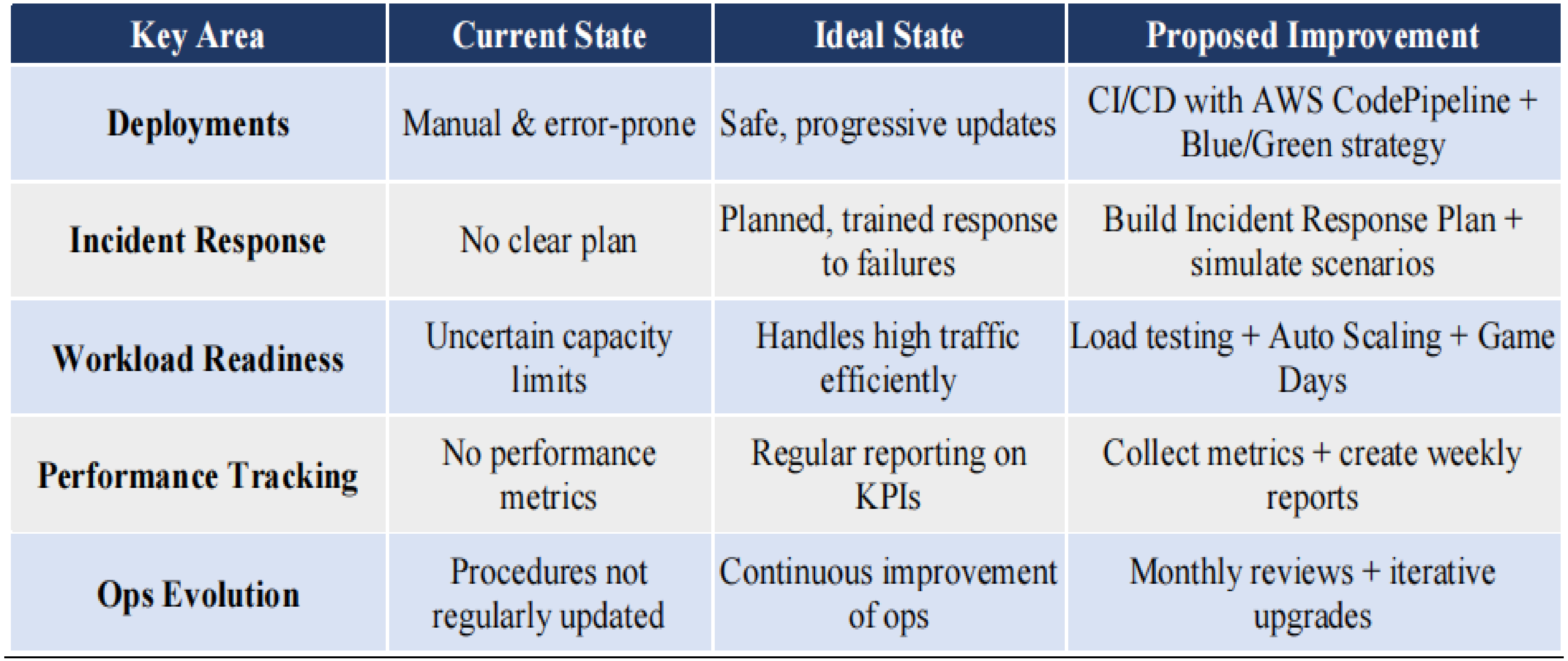


Operational Excellence pillar

Key Area	Current State	Ideal State	Proposed Improvement
System Monitoring	No centralized monitoring	Health checks for each service	Use AWS CloudWatch + X-Ray + Dashboard
Priority Setting	No clear prioritization framework	Priorities based on customer/business needs	Weekly review meetings to define focus areas
Org Structure	Disconnected departments	Integrated, cross-functional teams	Create shared channels + document workflows
Work Culture	Limited focus on innovation	Culture of improvement and collaboration	Host knowledge-sharing sessions + encourage feedback
Quality Testing	No pre-deployment testing	Automated quality checks	Create Staging environment + CI test automation



Operational Excellence pillar



Key Area	Current State	Ideal State	Proposed Improvement
Deployments	Manual & error-prone	Safe, progressive updates	CI/CD with AWS CodePipeline + Blue/Green strategy
Incident Response	No clear plan	Planned, trained response to failures	Build Incident Response Plan + simulate scenarios
Workload Readiness	Uncertain capacity limits	Handles high traffic efficiently	Load testing + Auto Scaling + Game Days
Performance Tracking	No performance metrics	Regular reporting on KPIs	Collect metrics + create weekly reports
Ops Evolution	Procedures not regularly updated	Continuous improvement of ops	Monthly reviews + iterative upgrades

Security pillar

Key Area	Current State	Ideal State	Proposed Improvement
Workload Security	No centralized control or auditing for workload access.	Secure, least-privilege access with centralized management.	Implement IAM roles and policies, enforce least privilege, and enable CloudTrail logging.
Security Event Detection	Limited visibility into potential security breaches.	Real-time detection and response to security events.	Enable Amazon GuardDuty, set up AWS Config rules, and integrate with AWS Security Hub.
Compute Protection	Basic security groups without fine-tuned rules or monitoring.	Hardened instances with least privilege and active monitoring.	Apply strict security group rules, use EC2 Image Builder, and automate patching with Systems Manager.
Identity and Access Management	Manual user account provisioning and inconsistent permission levels.	Automated user provisioning with fine-grained permissions.	Use AWS IAM Identity Center (SSO), attach permissions boundaries, and enforce MFA for all users.

Security pillar

Key Area	Current State	Ideal State	Proposed Improvement
Infrastructure Protection	Flat network architecture with minimal segmentation.	Isolated, well-defined segments with strict access rules.	Redesign VPCs using subnets, route tables, and security groups; implement network ACLs and AWS Firewall Manager.
Data Protection	Sensitive data stored without encryption; unclear data classification.	Data encrypted in transit and at rest with clear classification.	Enable KMS encryption, implement S3 bucket policies, and use Macie for data classification.
Incident Response	No formal incident response plan or playbooks.	Well-defined, tested incident response workflows.	Develop runbooks, use AWS Systems Manager Automation, and simulate incidents regularly.

Reliability Pillar

Key Area	Current State	Ideal State	Proposed Improvement
Service Quotas	No active tracking or alarms; risk of hitting limits unnoticed	Monitored via CloudWatch; proactively request quota increases	Integrate AWS Service Quotas with CloudWatch and set up alarms
Network Topology	Simple VPC design; likely single AZ; limited fault isolation	Multi-AZ/multi-region VPCs with public/private subnets	Redesign network with proper VPC segmentation and multi-AZ support
Service Architecture	Heavy reliance on EC2 with minimal managed services	Use of managed, scalable AWS services like ECS, Lambda, and RDS	Refactor services to use managed, decoupled architectures
Interaction Failure Prevention	No retries, timeouts, or circuit breakers	Use idempotent design, timeouts, retry logic, and circuit breakers	Add retry/backoff logic and implement circuit breakers
Interaction Failure Mitigation	No dead-letter queues or fallback mechanisms	DLQs, alerts, and fallback flows for failed messaging	Create DLQs for all SQS queues and enable CloudWatch alerts
Monitoring Resources	Basic CloudWatch metrics only; no full observability	End-to-end monitoring using CloudWatch, X-Ray, and custom dashboards	Implement unified observability with logs, traces, and metrics
Demand Adaptability	Web front-end auto scales; backend components are static	Elastic scalability across all service tiers	Enable Auto Scaling for backend EC2 and tune based on usage patterns

Reliability Pillar

Key Area	Current State	Ideal State	Proposed Improvement
Change Implementation	Manual deployments; no automated CI/CD pipeline	Automated deployments with rollback and approval stages	Implement CI/CD using AWS CodePipeline, CodeBuild, and IaC
Data Backup	Only tapes for flight data; no automated backups for live systems	Automated and versioned backups across services	Enable RDS automated backups, S3 versioning, and DR drills
Fault Isolation	Single AWS account; minimal separation between environments	Environment and service isolation with strict IAM controls	Separate workloads into different AWS accounts/VPCs and apply least privilege IAM
Component Failure Resilience	No auto-healing or failover setup	Graceful degradation and self-healing infrastructure	Configure EC2 Auto Recovery, load balancing, and multi-AZ/multi-region failover
Reliability Testing	No fault injection or chaos testing	Regular game days and chaos engineering tests	Schedule game days and implement automated failure scenarios
Disaster Recovery	Tape-based backups only; no formal DR strategy	Documented and tested DR plans with replication	Develop and test cross-region DR strategy with clear RTO/RPO goals

Performance Efficiency pillar

Key Area	Current State	Ideal State	Proposed Improvement
Architecture Selection	Manual decisions without performance benchmarking	Use of AWS Well-Architected Tool and performance testing	Adopt performance testing and AWS Well-Architected Tool reviews
Compute Solution	Primarily EC2; no ARM-based or serverless use	Use right-sized, diversified compute including Graviton and Lambda	Evaluate compute needs and migrate workloads to cost-effective, performant options
Storage Solution	General-purpose S3 and EBS usage	Tiered storage based on access patterns (e.g., S3 IA, S3 Glacier)	Implement intelligent-tiering and lifecycle policies