AWS Account Structure Documentation

# 1. Introduction

This document describes the account separation strategy adopted for our Amazon Web Services (AWS) environments.  
We maintain three AWS accounts:  
- Non-Production (UAT) – for development, testing, staging, and select legacy workloads.  
- Production (Prod) – for business-critical, customer-facing workloads.  
- QA – provisioned but not currently in active use.  
  
The purpose of this separation is to improve security, reduce risks, and allow clear governance, monitoring, and cost management between environments.

# 2. Purpose of Each Account

## 2.1 Non-Production (UAT) Account

The Non-Production (UAT) account is a sandboxed environment where new features and workloads are developed and validated.  
- Primary Use Case: Feature development, integration testing, and user acceptance testing (UAT).  
- Access Controls: Developers and QA engineers have broader privileges for resource provisioning and modifications.  
- Data Sensitivity: Primarily contains sanitized or synthetic datasets. However, in certain cases, controlled subsets of production data may also be present (see Section 2.3).  
- Cost Considerations: Designed for flexibility and cost savings; workloads may be scaled down or shut off outside business hours.

## 2.2 Production Account

The Production account hosts live, mission-critical workloads.  
- Primary Use Case: Running applications and services used by end users or business teams.  
- Access Controls: Strict “least privilege” IAM policies, with changes restricted to approved administrators.  
- Data Sensitivity: Contains sensitive business data and must comply with regulatory requirements (e.g., GDPR, HIPAA, SOC 2).  
- Cost Considerations: Optimized for reliability and performance; resources are always-on and designed for high availability.

## 2.3 Legacy Applications in Non-Production

In addition to modern workloads, the Non-Production (UAT) account also hosts several legacy applications that were originally deployed without a dedicated development setup. As a result, UAT sometimes serves as both the development and testing environment for these workloads.  
  
Challenges:  
- No Dev Environment: Developers make changes directly in UAT, which also serves as the validation environment.  
- Increased Risk: Shared development and testing increases the likelihood of service instability.  
- Limited Automation: Legacy applications are not fully aligned with Infrastructure as Code and pipeline practices.  
- Use of Production Data in UAT: In specific cases, limited subsets of production data are copied into UAT for validation or troubleshooting.  
  
Mitigation and Controls:  
- Business and Security Approval Required: Production data is only introduced into UAT after explicit review and approval from both the Business team and the Security team.  
- Data Handling Restrictions: Only the minimum dataset required for validation is copied, with masking/anonymization applied wherever possible.  
- Access Restrictions: Access to production-like datasets in UAT is limited to authorized engineers.  
- Audit and Monitoring: All data transfers are logged and monitored for compliance.  
  
Ongoing Collaboration:  
The Cloud Team is working closely with AWS and the Security Team to:  
- Establish stricter data handling policies for UAT.  
- Strengthen monitoring and alerting when production data is copied.  
- Define a long-term roadmap to re-architect legacy applications so they no longer require production data in non-production environments.

## 2.4 QA Account (Not in Active Use)

A separate QA account exists but is not actively used at this time.  
  
- Intended Purpose: Originally created to serve as an intermediate testing environment between UAT and Prod for regression testing, system integration testing (SIT), and performance validation.  
- Current Status:  
 \* The account is provisioned but not part of regular workflows.  
 \* No active deployments or workloads are running.  
 \* Most testing activities are currently consolidated within the UAT account.  
- Future Considerations:  
 \* The QA account may be repurposed in the future to separate functional testing from user acceptance testing (UAT).  
 \* Could also be used as a staging environment for critical releases requiring production-like validation.  
 \* Activation of this account would require alignment with the Cloud, Security, and Business teams to define governance, usage policies, and cost allocations.

# 3. Governance and Security Model

Identity and Access Management:  
- Centralized authentication via AWS IAM Identity Center or corporate SSO (e.g., Okta, Azure AD).  
- Role-Based Access Control (RBAC) applied consistently across all accounts.  
- Temporary session credentials replace static long-lived keys.  
  
Networking:  
- Each account has its own VPC with segregated public and private subnets.  
- VPC Peering or Transit Gateway is used if inter-account communication is required.  
- Security groups and NACLs enforce least-privilege network access.  
  
Security Services:  
- AWS GuardDuty, Security Hub, and AWS Config enabled in all accounts for continuous monitoring.  
- Centralized logging: CloudTrail, CloudWatch Logs, and VPC Flow Logs are stored in a designated logging bucket.  
- Separate KMS keys per account for encryption and isolation.

# 4. Deployment and Infrastructure Strategy

## 4.1 Current State

Currently, AWS CloudFormation is used for provisioning and managing infrastructure.  
- Templates are version-controlled in Git repositories.  
- Environment-specific differences are handled using CloudFormation parameters.  
- Updates are performed manually by operations teams through CloudFormation stack updates.  
- Provides traceability, consistency, and rollback capabilities.

## 4.2 In-Progress Enhancements

We are implementing CI/CD pipelines to automate deployments.  
- Pipeline Tooling: AWS CodePipeline and CodeBuild, integrated with Git.  
- Deployment Flow: New changes will be deployed first to UAT, validated, and then promoted to Prod.  
- Artifact Management: Versioned artifacts (e.g., Lambda packages, container images) will be stored in AWS CodeArtifact or ECR.  
- Future Benefits: Automation will improve speed, reduce manual errors, and ensure consistent deployments across environments.

# 5. Monitoring and Cost Management

Monitoring:  
- CloudWatch metrics, dashboards, and alarms set up separately per account.  
- CloudTrail captures all API activity and stores logs in centralized repositories.  
  
Cost Management:  
- AWS Budgets configured to track spend by account.  
- Cost allocation tags (e.g., Environment=UAT, Environment=Prod, Environment=QA) applied to resources.  
- Regular spend reviews conducted by the cloud operations team.

# 6. Benefits of Account Separation

- Security Isolation: Issues in UAT cannot directly impact production.  
- Cost Transparency: Each account’s costs are tracked independently.  
- Risk Reduction: Testing in UAT minimizes risk to live workloads.  
- Compliance Readiness: Account boundaries support external audits and governance.  
- Operational Efficiency: Teams work in parallel with clear environment boundaries.  
- Future Flexibility: The QA account provides an option to expand into a staging or performance testing environment.