**Assignment 2 Karamjit Judge N01477742**

**AWS Multi-Account Strategy Proposal for Unicorn Games**

Unicorn Games is a fast-growing game company that requires a secure, scalable, and well-structured AWS environment to support its development, testing and production workloads. A multi-account under AWS is recommended for the company to achieve isolation, security, cost effectiveness, and operational efficiency.  
  
 This proposal outlines the account structure, Justification, and security considerations for the multi-account strategy.

1. **The justification**

Why should Unicorn Games use multiple AWS Accounts?

|  |  |
| --- | --- |
| **Benefit** | **Description** |
| Security and Isolation | Prevents lateral movement in the case of a breach thus limiting the blast radius. |
| Cost and Budgeting | Clear cost tracking for each environment (Development, Testing, and Production) |
| Compliance and Governance | Easier enforcement of policies for each account (e.g. PCI and GDPR) |
| Operational Independence | Teams can manage their own environments without being able to interfere with other environments. |
| Service Quota Management | AWS service limits for things like EC2 instances are per account which reduces the risk of throttling. |

**Potential Challenges**

|  |  |
| --- | --- |
| **Challenge** | **Mitigation Strategy** |
| Increased Management Overhead | We can mitigate this by using AWS Organizations, SCPs, and AWS Control Tower for governance. |
| Cross-Account Access Complexity | Implement IAM Roles and AWS SSO for secure access. |
| Cost Tracking Complexity | Use AWS Cost Explorer and tagging strategies to deal with this. |

**Proposed AWS Account Structure**

AWS Organization Master (A Management Account)

Purpose:

* Central Governance for all AWS accounts
* It will manage the Service Control Policies, AWS Organizations, and consolidated billing.

Security Role:

* Enforces the guardrails (e.g. denying certain regions and restricting root user actions).
* Hosts AWS SSO for centralized access management.

Pros:

* Single point of control for security policies
* Simplifies compliance auditing.

Cons:

* Requires strict access control to prevent misuse

Software Development Lifecycle Accounts:

Development Account

Purpose:

* Used by developers for feature development and initial testing.

Security Role:

* Enforces least privileged IAM policies
* Restricts Production Access

Pros:

* Isolates experimental changes from stable environments.

Cons:

* Requires strict management to avoid drift.

Testing/Quality Assurance Account

Purpose:

* Will be used for automated and manual testing before production deployment.

Security Role:

* No direct production access
* Uses automated security scanning tools like AWS Inspector

Pros:

* Ensures stability before production

Cons:

* Needs synchronization with development and production environments

Staging/Pre-Production Account

Purpose:

* Mirrors production for final validation

Security Role:

* Tightly controlled, near-production security policies.

Pros:

* Reduces deployment risks

Cons:

* Requires frequent updates to match production versions.

Production Account

Purpose:

* Hosts live customer-facing game services.

Security Role:

* Highest security control because of the ability to use Multi Factor Authentication enforcement and encryption at rest
* Logging and monitoring using AWS GuardDuty and CloudTrail

Pros:

* Minimizes risk of unauthorized changes

Cons:

* Strict change approval process may slow down deployments.

**Extra Credit**

DevOps Account  
Purpose:

* Centralizes CI/CD pipelines (AWS CodePipeline and Jenkins).
* Manages infrastructure-as-code (IaC) (Terraform and AWS CDK).

Security Role:

* Grands deployment permissions to other accounts via IAM Cross-Account Roles
* Enforces automated security checks in pipeline

Pros:

* Reduces risk of misconfigurations in Product
* Standardized deployments

Cons:

* Requires careful IAM role design to avoid over-permissions.

**Additional accounts (I’m too tired to flesh these out as much as the previous accounts)**

|  |  |  |
| --- | --- | --- |
| **Account Type** | **Purpose** | **Security Benefit** |
| Security and Logging | Centralized CloudTrail, GuardDuty, and Security Hub. | Single source for threat detection |
| Shared Services | Hosts VPC endpoints, DNS (Route 53) and VPN. | Reduces Duplicate resources |
| Analytics | S3 buckets, ETL (Glue), BI (QuickSight). | Isolates sensitive data processing |
| Sandbox | Experimental testing without the fear of affecting other accounts. | Contains untrusted workloads. |

**Recommended next steps:**

1. Implement AWS Organizations with SCPs
2. Deploy AWS Control Tower for automated guardrails
3. Set up AWS SSO for centralized access
4. Enforce mandatory tagging for cost tracking
5. Automate security checks in CI/CD pipelines.

**Conclusion:**

Implementing a multi-account AWS strategy provides Unicorn Games with a scalable, secure, and well-governed cloud environment, aligning with the industry’s best practices for gaming companies. This approach ensures isolation between the environments, reducing risks created by unauthorized access, misconfigurations, and compliance violations. By structuring accounts according to development lifestyle stages and separating security, logging, and shared services, Unicorn Games can achieve fine-tuned control over permissions, cost tracking, and auditability. Therefore, by adopting a AWS Multi-Account Strategy, Unicorn Games will not only meet current operational needs but will also future proof its cloud infrastructure for upcoming game launches and global scaling.

**Sources**

**Works Sited**

Amazon Web Services. *AWS Organizations User Guide*. AWS Documentation, 2023, <https://docs.aws.amazon.com/organizations/latest/userguide/>.

---. *AWS Well-Architected Framework: Security Pillar*. AWS Documentation, 2023, <https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html>.

---. "Best Practices for Organizational Units (OUs)." *AWS Whitepapers*, 2023, <https://docs.aws.amazon.com/whitepapers/latest/organizing-your-aws-environment/organizing-your-aws-environment.html>.

---. *AWS Control Tower User Guide*. AWS Documentation, 2023, <https://docs.aws.amazon.com/controltower/latest/userguide/>.

---. *AWS Tagging Strategies*. AWS Answers, 2023, <https://aws.amazon.com/answers/account-management/aws-tagging-strategies/>.

---. *AWS Multi-Account Security Strategy*. AWS Whitepaper, 2023, <https://d1.awsstatic.com/whitepapers/aws-multi-account-security-strategy.pdf>.

---. "Setting Up a Multi-Account AWS Environment." *AWS Blog*, 2023, <https://aws.amazon.com/blogs/aws/setting-up-a-multi-account-aws-environment/>.

---. "Security Best Practices for Multiple AWS Accounts." *AWS Security Blog*, 2023, <https://aws.amazon.com/blogs/security/tag/multi-account/>.

---. "CI/CD Best Practices for DevOps on AWS." *AWS DevOps Blog*, 2023, <https://aws.amazon.com/blogs/devops/category/devops/>.

---. *AWS Deployment Framework (ADF) Documentation*. AWS Solutions, 2023, <https://aws.amazon.com/solutions/implementations/aws-deployment-framework/>.

---. *AWS Cost Optimization Best Practices*. AWS Cost Management, 2023, <https://aws.amazon.com/aws-cost-management/aws-cost-optimization/>.