AWS 백서

게임 산업 렌즈

게임 산업 렌즈: AWS 백서

Table of Contents

Games Industry Lens - AWS Well-Architected 프레임워크

게시일: 2025년 12월 9일(문서 수정)

AWS Well-Architected Framework는 클라우드 아키텍트가 애플리케이션 및 워크로드를 위한 안전하고 성능이 뛰어나며 복원력이 뛰어나고 효율적인 인프라를 구축할 수 있도록 지원합니다. 운영 우수성, 보안, 신뢰성, 성능 효율성, 비용 최적화 및 지속 가능성이라는 6가지 원칙을 기반으로 하는 Well-Architected는 고객과 AWS 파트너가 아키텍처를 평가하고, 위험을 해결하고, 비즈니스 가치를 제공하는 설계를 구현할 수 있는 일관된 접근 방식을 제공합니다.

이 렌즈에서는에서 게임 워크로드를 설계, 배포 및 설계하는 방법에 중점을 둡니다 AWS 클라우드. Well-Architected Framework를 적용하는 데 도움이 되는 구성 요소를 정의하고, 일반적인 워크로드 시나리오를 살펴보고, 설계 원칙을 간략하게 설명합니다. AWS Well-Architected Framework 백서의 모범 사례와 질문을 고려하여 아키텍처 설계를 시작하는 것이 좋습니다. 이 문서는 게임 업계 고객을 위한 보충 모범 사례를 제공합니다.

이 렌즈는 전 세계 게임 산업 개발자 및 게시자와 협력한 경험을 기반으로 클라우드에서 게임을 구축하고 운영하는 고유한 특성을 해결하기 위한 모범 사례를 지정합니다. 글로벌 플레이어 수요의 변동에 맞게 비용을 최적화하고 확장할 수 있도록 환경을 설계하고 운영하는 방법에 대한 지침을 제공합니다. 또한이 렌즈는 게임 인프라를 보호하고 긍정적인 플레이어 경험을 제공하도록 성능을 조정하기 위한 지침을 제공합니다.

이 문서는 최고 기술 책임자(CTOs), 게임 스튜디오 기술 책임자, 아키텍트, 개발자 및 운영 팀원과 같은 기술 역할 담당자를 대상으로 합니다. 이 문서를 읽은 후에는 게임용 아키텍처를 설계할 때 사용할 AWS 모범 사례와 전략을 이해하게 됩니다.

렌즈 가용성

사용자 지정 렌즈는에서 제공하는 모범 사례 지침을 확장합니다 AWS Well-Architected Tool.를 AWS WA Tool 사용하면 사용자 지정 렌즈를 생성하거나 공유된 다른 사용자가 생성한 렌즈를 사용할 수 있습니다.

게임 워크로드 검토를 시작하려면 퍼블릭 AWS Well-Architected 사용자 지정 GitHub 렌즈 GitHub 리포지토리에서 Games Industry Lens를 다운로드하여 로 가져옵니다. AWS Well-Architected Tool

설계 원칙

AWS Well-Architected 프레임워크는 다음과 같은 일반적인 설계 원칙을 식별하여 게임 워크로드에 적합한 클라우드 설계를 지원합니다.

- 플레이어 행동 및 사용 패턴을 이해하여 게임을 발전시키고 플레이어를 보호합니다. 게임을 지속적으로 개선하고 플레이어 경험을 효과적으로 관리하려면 플레이어가 게임 자체 및 나머지 플레이어와 상호 작용하는 방식을 파악하는 것이 중요합니다. 이를 통해 게임을 개선하고, 비용을 관리하고, 플레이어 경험에 위험을 초래하는 무단 사용 활동을 모니터링하고 이에 대응하는 방법을 이해할 수 있습니다.
- 게임 운영을 간소화하고 개발 속도를 높이는 기술 사용: 속도를 개선하고 플레이어에게 새로운 기능과 개선 사항을 제공하는 운영 오버헤드를 줄일 수 있는 기술 채택의 우선순위를 정합니다. 게임은 히트 중심이며 플레이어가 고려해야 할 선택 사항이 많으므로 빠르게 움직이고 변화에 적응하는 것이 게임의 성공에 매우 중요합니다. 자체 소프트웨어를 운영하는 데 익숙한지 아니면 AWS, AWS Partners 또는 둘 다에서 유사한 관리형 서비스를 채택하고 싶은지 생각해 보세요.
- 아키텍처를 최적화하여 실제 플레이어 경험을 반영하는 지표를 개선합니다. 시간이 지남에 따라 아키텍처를 조정하고 발전시킬 때 이러한 개선 및 변경이 플레이어 경험에 어떤 영향을 미칠지 생각해 보세요. 게임 워크로드는 장애의 영향을 견디고 최소화하여 게임 플레이에 대한 광범위한 중단을 차단할 수 있어야 합니다. 서로 크게 의존하지 않는 게임 기능과 시스템을 분리하여 장애의 영향을 줄이고 플레이어가 문제에 미치는 영향을 격리해야 합니다.
- 피크 플레이어 동시성을 충족하고 필요에 따라 동적으로 규모를 조정하도록 인프라 설계: 인프라는 플레이어 수요에 맞게 규모를 조정하도록 설계되어야 합니다. 플레이어 세션 동시성 및 로그인 수와 같은 지표를 사용하여 시스템에 과부하가 걸리기 전에 선제적으로 규모를 조정할 수 있습니다. CPU 및 메모리 소비와 같은 대응 시스템 사용률 지표를 사용하여 시스템에 과부하가 걸린 후 규모를 조정할 수 있습니다. 인프라를 동적으로 확장하면 게임 운영 비용을 줄일 수 있습니다.
- 런북을 구현하여 게임 운영 개선: 반복되는 게임 운영 작업을 일관되게 관리하려면 운영 런북을 사용해야 합니다. 플레이어 보고서 조사 및 응답, 새 시즌 출시 및 게임 콘텐츠 릴리스와 같은 예상 대규모 이벤트에 대비하기 위한 인프라 사전 규모 조정 활동 관리, 일반적인 게임 유지 관리 활동 처리와 같은 일반적인 게임 운영 워크플로에 런북이 있어야 합니다.

시나리오

이 섹션에서는 게임 아키텍처에서 일반적인 몇 가지 시나리오를 다룹니다. 각 시나리오에는 설계를 주도하는 일반적인 특성과 예제 참조 아키텍처 다이어그램이 포함되어 있습니다.

시나리오

실시간 동기식 게임 플레이를 위한 게임 호스팅

실시간 동기식 게임 플레이를 사용하면 두 명 이상의 플레이어가 게임에 동시에 참여하고 상호 작용할 수 있습니다.이 경우 연결된 플레이어 간에 게임 플레이 상태가 공유되어 실시간 경험에 최대한 가깝게 생성할 수 있습니다. 동기 게임의 예로는 1인칭 슈팅 게임, 대규모 멀티플레이어 온라인 게임(MMOG), 스포츠 및 액션 게임 또는 두 명 이상의 플레이어를 연결하여 거의 실시간으로 플레이 경험을 공유해야 하는 온라인 게임이 있습니다.

실시간 동기식 게임 플레이 아키텍처의 특성은 다음과 같습니다.

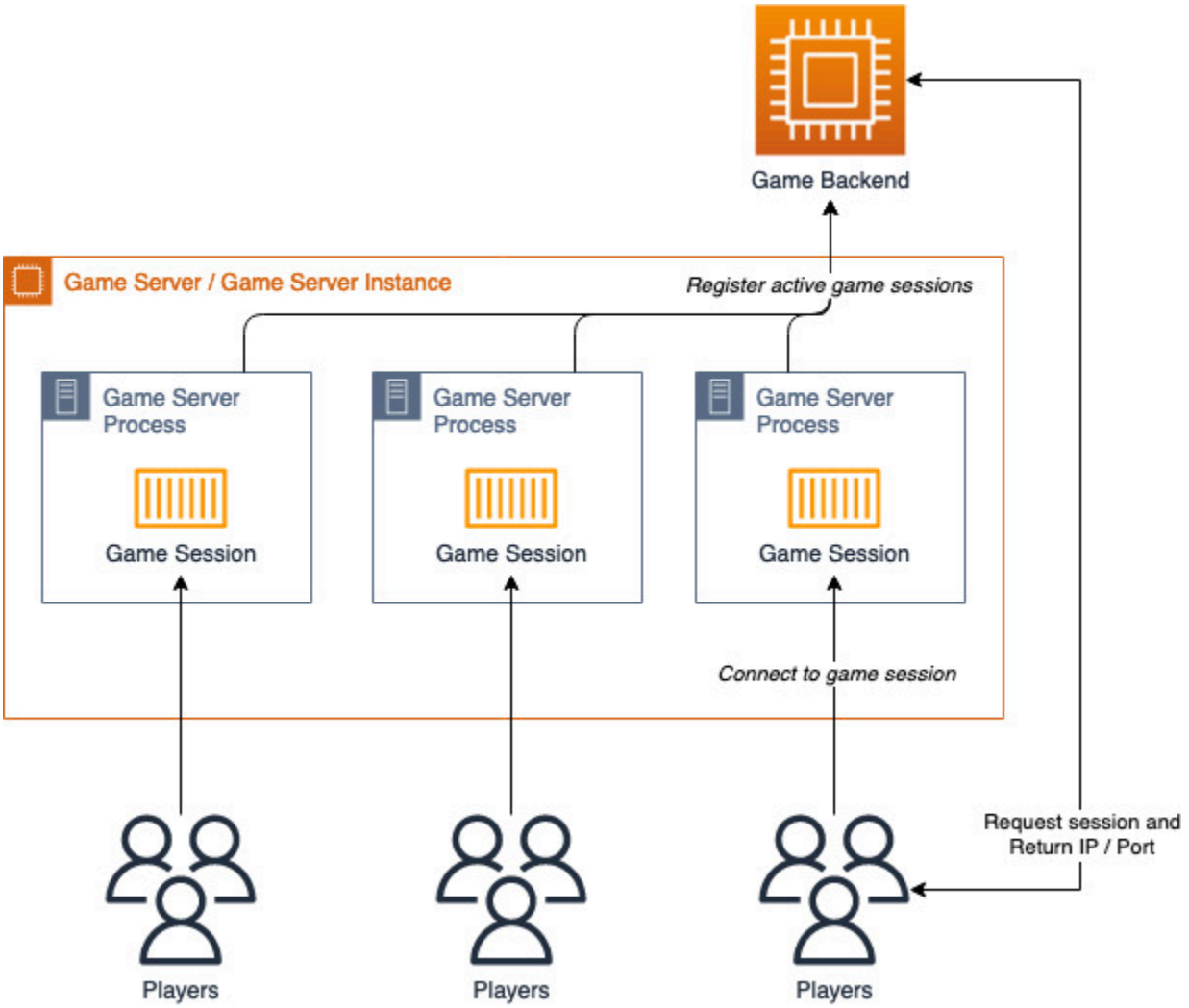
- 일부 게임은 전용 서버에서 실행되는 게임 서버 프로세스를 통해 게임 세션으로 호스팅될 수 있습니다. 일부 게임은 NAT(STUN)용 경량 세션 순회 유틸리티 또는 NAT(TURN) 서버 주변의 릴레이를 사용한 순회를 사용하는 P2P-like 아키텍처를 사용할 수 있습니다. 관련된 서버 유형에 관계없이 게임 서버는 여러 데이터 센터와 AWS 리전 전 세계에서 호스팅됩니다.
- 게임 클라이언트는 게임 백엔드 시스템에서 호스팅되는 중앙 집중식 매치메이킹 서비스에서 매치를 요청하거나 사전 정의된 사용 가능한 게임 서버 목록에서 매치를 선택하여 게임 세션에 참여할 수 있습니다. 게임 클라이언트에는 연결할 IP 주소와 포트가 제공됩니다.
- 많은 동기식 게임은 1인칭 슈팅 게임 및 대규모 멀티플레이어 온라인 게임과 같이 지연 시간에 민감합니다. 지연 효과를 최소화하기 위해 되감기 및 시간 확장과 같은 알고리즘이 포함될 수 있지만 지연 시간이 긴 상황에서 플레이어에게 발생할 수 있는 지연 경험을 줄이기 위해 신중하게 측정되고 최적화되는 사전 정의된 지연 시간 허용 오차도 있을 수 있습니다. 이 지연 시간 정보는 게임 클라이언트를 계측하여 사용 가능한 게임 서버를 ping AWS 리전 하여 지연 시간, 네트워크 지터 및 게임 플레이 경험에 대한 기타 중요한 지표와 같은 지표를 캡처함으로써 결정됩니다. 이러한 지표는 게임 백엔

드 시스템의 중앙 지표 수집 서비스로 전송되므로 라이브 작업 스트림이 게임 상태를 모니터링할 수 있습니다. 매치메이킹 프로세스 중에 게임 클라이언트는 매치를 요청할 때 현재 지연 시간 데이터를 요청 파라미터 중 하나로 제공하며, 매치메이킹 서비스는 플레이어를 호스팅할 게임 서버를 선택할 때 해당 지연 시간 데이터를 변수 중 하나로 사용할 수 있습니다.

- 일반적으로 게임 플레이는 여러 프로토콜을 통해 수행됩니다(예: 매치메이킹, 인증 및 HTTPS를 사용하는 기타 클라이언트-서버 트래픽과 함께 더 빠른 UDP 기반 메시징을 사용하는 게임 서버).
- 게임 서버는 알고리즘과 설계를 사용하여 변환 스트리밍, 델타 및 데이터 압축과 같은 클라이언트-서버 트래픽을 최소화합니다.
- 게임 서버는 악의적인 활동의 빈번한 대상이며 같은 DDoS 보호 솔루션으로 보호해야 합니다 AWS Shield Advanced.

게임 서버 프로세스

다음 다이어그램은 일반적인 게임 서버 아키텍처를 보여줍니다. 게임 서버 인스턴스와 게임 세션을 호스팅하는 게임 서버 프로세스 간의 논리적 관계를 설명합니다.



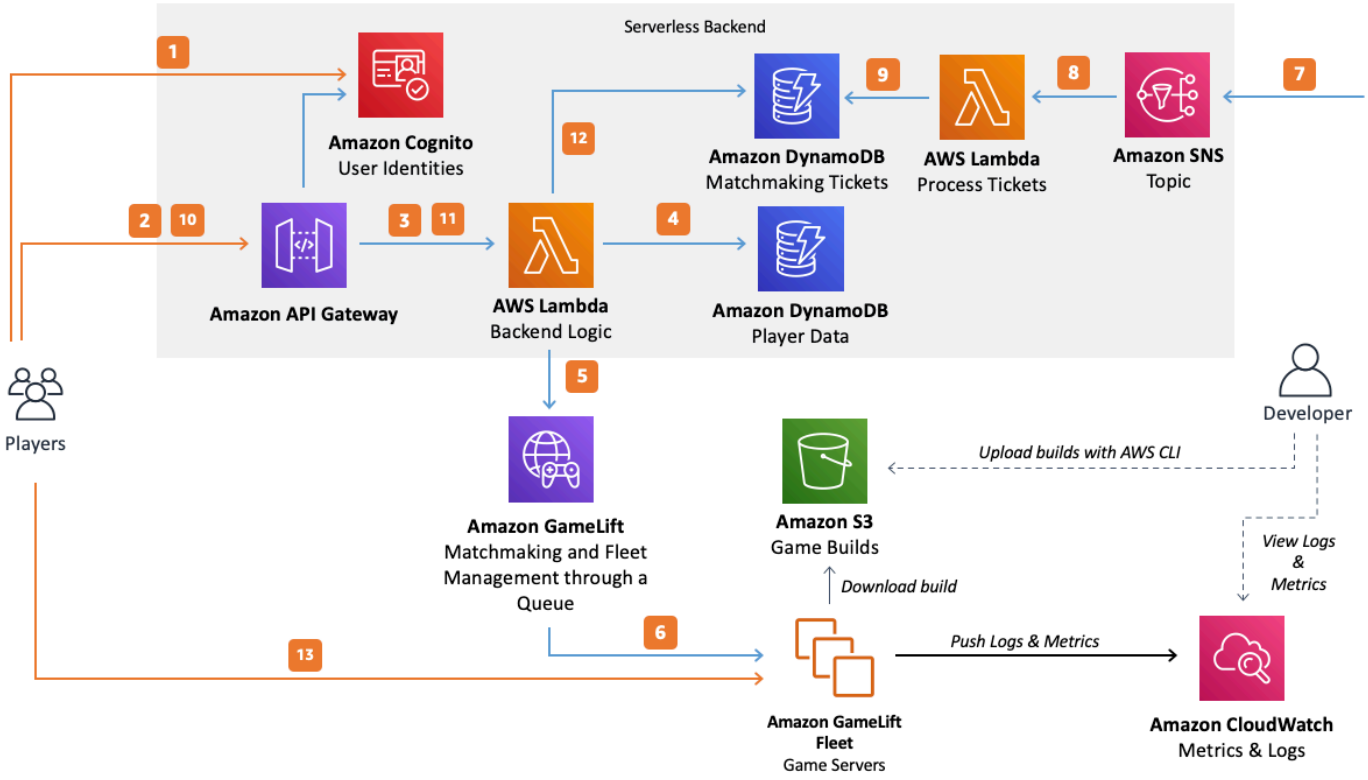
논리적 게임 서버 아키텍처

- Amazon EC2 인스턴스는 게임 서버 인스턴스라고도 하는 게임 서버로 사용됩니다. 게임 서버는 하나 이상의 게임 서버 프로세스를 호스팅하며, 각 프로세스는 게임 서버 빌드의 복사본을 실행하고 있습니다. 일반적으로 여러 게임 서버 프로세스가 게임 서버 인스턴스에서 실행되어 컴퓨팅 리소스를 효율적으로 활용하고 비용을 절감합니다. 게임 세션이 활성 상태이고 플레이어 세션을 호스팅할 준비가 되면 플레이어를 호스팅하는 데 사용할 수 있도록 게임 백엔드(일반적으로 매치메이킹 서비스)로 상태가 업데이트됩니다.
- 게임 백엔드는 플레이어가 플레이에 연결할 수 있도록 게임 세션을 호스팅하는 IP 주소와 서버 포트를 플레이어에 제공할 수 있습니다.

서버리스 백엔드를 사용한 세션 기반 게임 서버 호스팅

게임용 아키텍처를 개발할 때는 필요한 기능과 소유할 준비가 된 운영 관리 오버헤드 수준을 고려하세요. 운영 편의성과 유연성 간에 최상의 균형을 맞추기 위해 클라우드 공급자의 관리형 서비스를 사용하여 게임을 빌드할 수 있습니다. 관리형 서비스를 사용하면 자체 사용자 지정 게임 기능을 개발하고 사용자 지정하는 동시에 인프라 배포 및 관리 부담을 줄일 수 있습니다.

세션 기반 멀티플레이어 게임을 호스팅하려면 게임 서버 프로세스를 호스팅하기 위한 서버 인프라와 매치메이킹 및 세션 관리를 위한 확장 가능한 백엔드가 있어야 합니다. 다음 참조 아키텍처는 Amazon GameLift 관리형 호스팅과 서버리스 백엔드를 사용하여 세션 기반 게임을 관리하는 방법을 보여줍니다.



세션 기반 게임을 위한 Amazon GameLift 관리형 호스팅

다이어그램은 플레이어를 GameLift 관리형 게임 호스팅에서 실행되는 게임에 참여시키는 프로세스를 설명합니다. 여기에는 다음 단계가 포함됩니다.

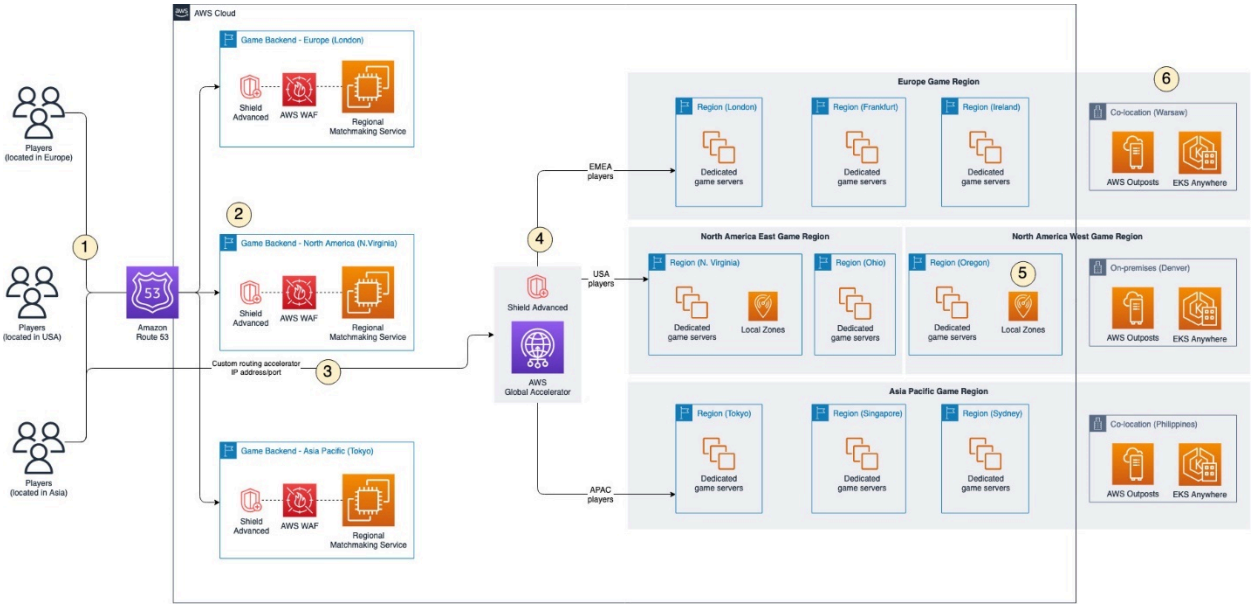
1. 게임 클라이언트는 Amazon Cognito 자격 증명 풀에서 Amazon Cognito 자격 증명을 요청합니다. 선택적으로 외부 자격 증명 공급자에 연결할 수 있습니다.
2. 게임 클라이언트는 임시 액세스 자격 증명을 수신하고 Amazon Cognito 자격 증명으로 요청에 서명하여 Amazon API Gateway를 통해 게임 세션을 요청합니다. Amazon Cognito

3. API Gateway는 AWS Lambda 함수를 호출합니다.
4. Lambda 함수는 Amazon DynamoDB 테이블에서 플레이어 데이터를 요청합니다. Amazon Cognito 자격 증명은 요청 컨텍스트 데이터에 인증된 자격 증명이 제공되므로 올바른 플레이어 데이터를 안전하게 요청하는 데 사용됩니다.
5. Lambda 함수는 추가 정보(예: 플레이어 스킬 레벨)를 위해 올바른 플레이어 데이터를 사용하여 GameLift FlexMatch 매치메이킹을 통해 매치를 요청합니다. JSON 기반 구성 문서를 사용하여 FlexMatch 매치메이킹 구성을 정의할 수 있습니다. 게임 클라이언트는 다양한 리전에서 서버 엔드 포인트를 ping하여 지연 시간 지표를 생성할 수 있으며 지연 시간 데이터를 사용하여 지연 시간 기반 매치메이킹을 지원할 수 있습니다.
6. FlexMatch는 리전에 적절한 지연 시간을 가진 적절한 플레이어 그룹을 매칭한 후 GameLift 대기열을 통해 게임 세션 배치를 요청합니다. 대기열에는 등록된 리전 위치가 하나 이상인 플릿이 포함되어 있습니다.
7. 세션이 플릿 위치 중 하나에 배치되면 이벤트 알림이 Amazon SNS 주제로 전송됩니다.
8. Lambda 함수는 Amazon SNS 이벤트를 수신하고 처리합니다.
9. Amazon SNS 메시지가 MatchmakingSucceeded 이벤트인 경우 Lambda 함수는 서버 포트 및 IP 주소를 사용하여 DynamoDB에 결과를 기록합니다. TTL(time-to-live) 값은 더 이상 필요하지 않을 때 매치메이킹 티켓이 DynamoDB에서 삭제되도록 하는 데 사용됩니다.
10. 게임 클라이언트는 API Gateway에 서명된 요청을 보내 특정 간격으로 매치메이킹 티켓의 상태를 확인합니다.
11. API Gateway는 매치메이킹 티켓 상태를 확인하는 Lambda 함수를 호출합니다.
12. Lambda 함수는 DynamoDB를 확인하여 티켓이 성공했는지 확인합니다. 성공하면 Lambda 함수는 IP 주소, 포트 및 플레이어 세션 ID를 클라이언트로 다시 전송합니다. 티켓이 실패하면 Lambda 함수는 매치가 준비되지 않았음을 선언하는 응답을 보냅니다.
13. 게임 클라이언트는 백엔드에서 제공하는 포트 및 IP 주소를 사용하여 TCP 또는 UDP를 사용하여 게임 서버에 연결합니다. 플레이어 세션 ID를 게임 서버로 전송하고 게임 서버는 Amazon GameLift Server SDK를 사용하여 이를 검증합니다.

또는 Amazon GameLift와 함께 API Gateway WebSockets를 사용하도록 이전 아키텍처를 수정할 수 있습니다. 이 접근 방식에서는 WebSocket 기반 구현을 사용하여 게임 클라이언트와 게임 백엔드 서비스 간의 통신이 이루어집니다. 이 구현을 사용하면 게임 백엔드 Lambda 함수가 폴링 모델을 구현하는 대신 WebSocket을 통해 게임 클라이언트에 대한 서버 측 메시지를 시작할 수 있습니다.

지연 시간이 짧은 게임을 위한 다중 리전 및 하이브리드 아키텍처

이 섹션에서는 지연 시간이 짧은 게임을 위한 다중 리전 및 하이브리드 아키텍처를 설명합니다.



전 세계에 배포된 네트워크 가속화 및 게임 서버로 지연 시간 단축

1. 전 세계적으로 사용 가능한 게임의 플레이어는 어디서나 시작할 수 있습니다. 플레이어가 게임 세션 또는 매치를 요청하면 게임 클라이언트는 Amazon Route 53에 등록된 게임 백엔드 서비스에 요청을 보냅니다. Route 53 지연 시간 기반 라우팅을 사용하여 플레이어를 사용 가능한 가장 가까운 게임 백엔드로 라우팅할 수 있습니다.
2. 게임 백엔드는 플레이어 모집단과 가장 AWS 리전 가까운 여러에 배포됩니다. 각 게임 백엔드에는 게임 리전 전체에서 게임 세션을 찾는 리전 매치메이킹 서비스가 포함되어 있습니다. 플레이어의 매치메이킹 요청은 근처의 리전 매치메이킹 서비스에서 처리되지만 필요한 경우 매치메이킹 서비스는 플레이어를 다른 게임 리전의 게임 세션으로 라우팅할 수 있습니다. 이 작업은 복원력과 성능을 개선합니다. 또한 각 게임 백엔드 서비스는AWS WAF 및 AWS Shield Advanced 를 사용하여 계층 7 웹 필터링 및 봇 제어와 배포 서비스 거부(DDoS) 공격에 대한 보호를 제공합니다. 서버리스, 컨테이너, EC2 인스턴스 또는 자체 데이터 센터에서 게임 백엔드 서비스를 호스팅하는 등 게임 백엔드 서비스를 빌드하는 방법에 대한 다양한 옵션이 있습니다.
3. 네트워크 지연 시간과 지터를 줄여 플레이어의 경험을 개선하기 위해 사용자 지정 라우팅 액셀러레이터는 AWS 글로벌 네트워크를 사용하여 게임 클라이언트에서 게임 서버로의 트래픽 라우팅을 자동으로 최적화하는 AWS Global Accelerator를 사용하여 배포됩니다. Global Accelerator 리스너 포트를 게임 서버의 EC2 인스턴스 포트에 매핑하도록 사용자 지정 라우팅 액셀러레이터를 구성합니다. 게임 클라이언트는 프록시 역할을 하는 Global Accelerator IP 및 포트에 연결하고 플레이어를 게임 세션을 호스팅하는 올바른 게임 서버 IP 및 포트로 결정적으로 라우팅합니다.
4. 게임에는 북미 또는 아시아 태평양과 같이 지리적으로 서로 가까운 게임 서버 호스팅 위치 모음을 나타내는 플레이어 친화적인 논리적 게임 리전이 포함되어 있습니다. 지연 시간을 줄이고 지리적 범위를 늘리기 위해 다양한 게임 서버 호스팅 솔루션을 조합하여 플레이어 경험을 개선할 수 있습

니다. 이러한 위치는 완전한 기능을 갖추고 있으며 가장 큰 용량 공간을 포함하고 있으므로 가능한 AWS 리전 경우 사용을 우선시합니다.

5. AWS 로컬 영역을 사용하여 기존 호스팅 시설이 없거나를 AWS 리전 사용할 수 없는 서비스가 부족한 플레이어의 지리적 위치에서 게임 서버를 호스팅합니다.
6. 기존 온프레미스 데이터 센터 및 코로케이션 공급자 AWS Outposts 에 배포하여 완전 관리형 랙 및 랙 탑재 서버를 사용하여 각 배포 위치에서 원활한 컨트롤 플레인 및 관리 환경을 생성합니다. AWS Outposts 는 온프레미스 환경에 기존 서버 용량이 없는 경우에도 유용합니다. 그러나 기존 서버 인프라에서 실행되는 소프트웨어를 사용한 하이브리드 구현을 원하는 경우 Amazon EKS Anywhere 를 사용해야 합니다. 그러면 Amazon EKS에 다시 연결하여 자체 인프라에서 Kubernetes 클러스터를 생성하고 실행할 수 있습니다. 이를 통해 온프레미스의 Kubernetes 클러스터에 대한 일관된 콘솔 보기가 제공됩니다.

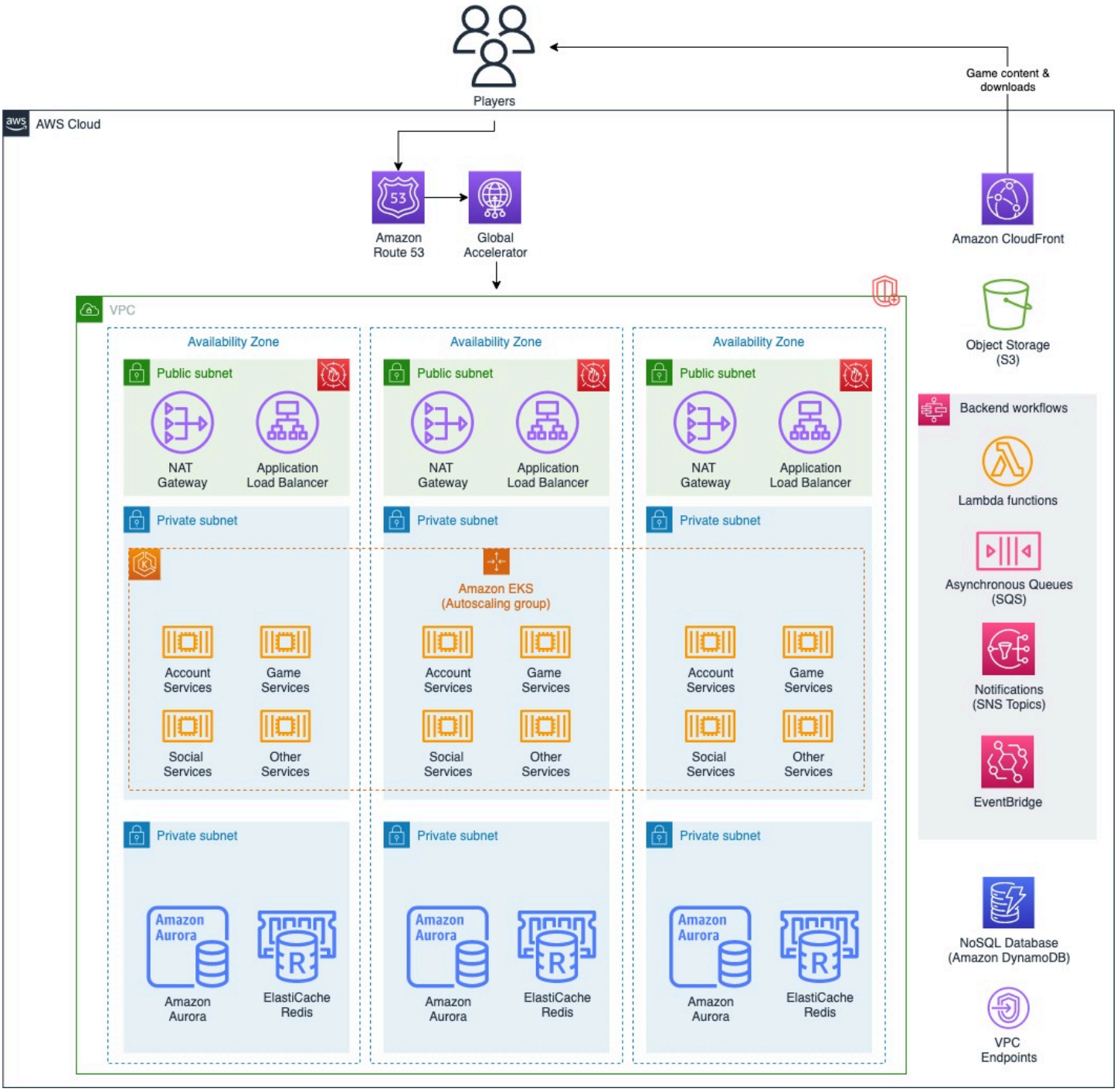
게임 백엔드

게임 백엔드는 게임 및 플레이어 상태를 관리하고 게임 경험을 지원하는 게임에 소셜 및 시스템 수준 기능을 통합하는 데 사용됩니다. 플레이어 프로필 관리, 항목 및 인벤토리 스토리지, 통계 및 리더보드는 게임 백엔드에서 호스팅되는 서비스의 예입니다.

게임 백엔드는 일반적으로 HTTPS를 사용하여 클라이언트가 액세스하는 REST APIs로 빌드됩니다. 그러나 게임 내 채팅 및 존재에 대한 클라이언트 알림과 같은 사용 사례에 양방향 채널을 제공하는 WebSockets과 같은 다른 접근 방식도 일반적입니다. 게임 백엔드는 인스턴스, 컨테이너 또는 서버리스 아키텍처 사용을 포함하여 다양한 배포 아키텍처를 사용하여 배포할 수 있습니다.

컨테이너 기반 게임 백엔드 아키텍처

이 섹션에서는 컨테이너 기반 게임 백엔드 아키텍처를 간략하게 설명합니다.



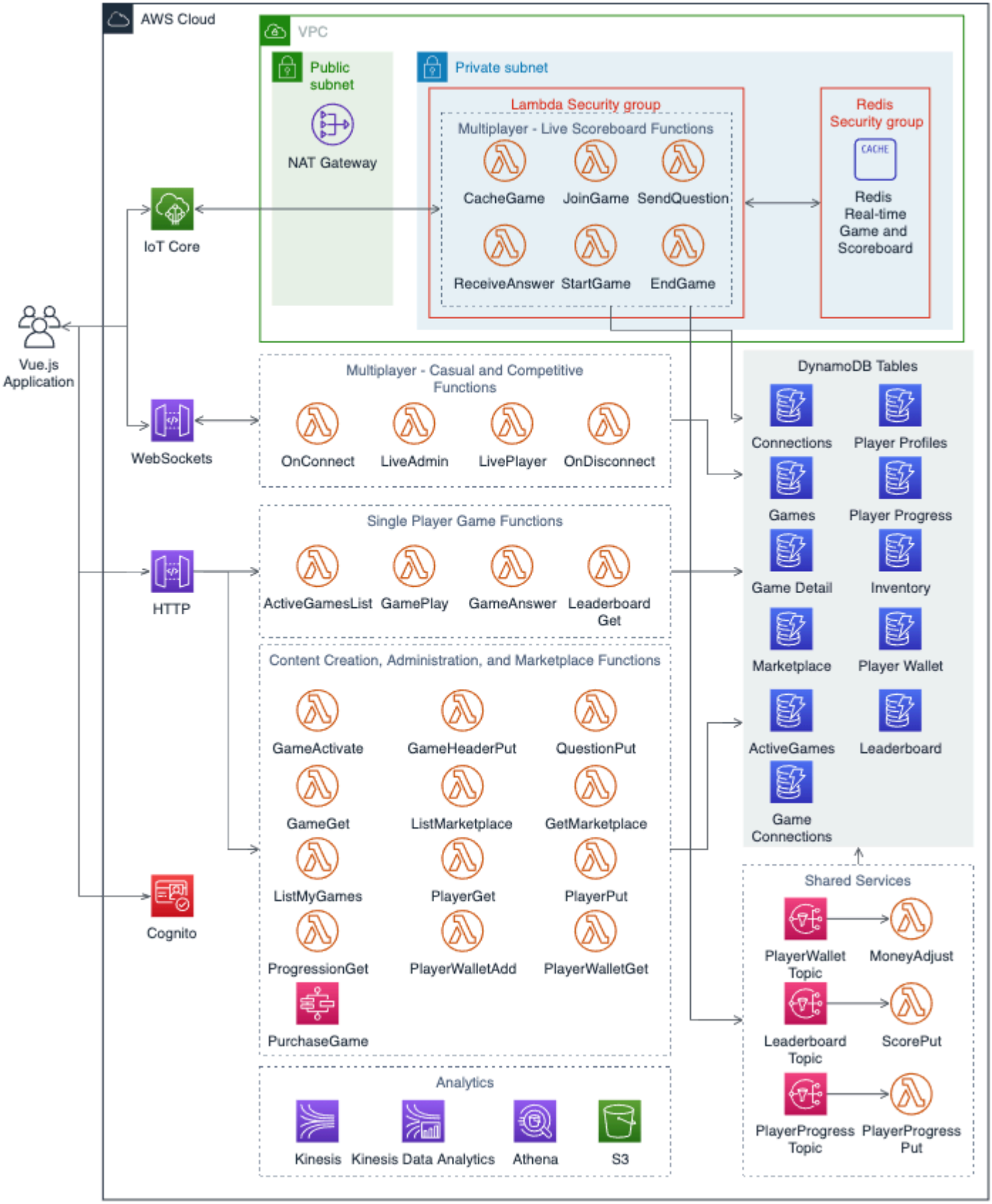
컨테이너를 사용하여 게임 백엔드 호스팅

- 플레이어는 게임 시스템, 디지털 스토어 프런트 또는 Amazon CloudFront와 같은 콘텐츠 전송 네트워크(CDN)에서 직접 다운로드를 통해 배포할 수 있는 게임 클라이언트 소프트웨어를 사용하여 게임에 액세스합니다. CDNs 엣지 로케이션에서 캐싱을 제공하여 콘텐츠를 다운로드하는 사용자의 성능을 가속화합니다. 예를 들어 CloudFront를 사용하여 게임 클라이언트 소프트웨어를 플레이어와 게임 자산 및 기타 콘텐츠에 배포할 수 있습니다.

- AWS Global Accelerator는 플레이어 게임 클라이언트에서 로드 밸런서로 트래픽을 라우팅하고 다중 리전 및 장애 조치를 위해 리전 간에 트래픽을 라우팅하기 위한 트래픽 가속화 및 사용자 지정 가능한 제어를 제공합니다. 게임 백엔드 REST APIs의 사용자 지정 도메인 이름은 트래픽을 Global Accelerator 엔드포인트로 라우팅하도록 Amazon Route 53에 구성됩니다. 다이어그램에 표시되지 않은는 액셀러레이터 및 게임 백엔드에 대한 추가 DDoS 완화를 제공할AWS Shield Advanced 수 있습니다.
- NAT Gateway 및 Application Load Balancer는 게임 백엔드에서 사용하는 각 가용 영역의 퍼블릭 서브넷에 배포되어 리전에서 고가용성을 제공합니다.AWS WAF 는 Application Load Balancer에 배포되어 계층 7 웹 트래픽 필터링을 제공합니다.
- 게임 백엔드는 복원력을 위해 가용 영역에 분산된 프라이빗 서브넷의 Amazon EKS 클러스터에 배포된 개별 컨테이너 기반 마이크로서비스의 모음으로 호스팅됩니다. 자동 조정은 일반적으로 플레이어 수요와 상관관계가 있는 리소스 사용률을 기반으로 서비스 및 클러스터 노드의 용량을 동적으로 조정합니다. Cluster Autoscaler는 클러스터의 노드 수를 자동으로 조정하는 반면 Horizontal Pod Autoscaler는 클러스터에 배포된 포드를 자동으로 조정합니다.
- 게임 및 플레이어 데이터는 기본 노드와 복제본 노드 간의 복제를 통해 가용 영역의 프라이빗 서브넷에 배포되는 백엔드 데이터베이스 및 캐시에 저장됩니다. Amazon Aurora는 플레이어 프로필, 권한 및 게임 내 구매와 같은 사용 사례에 널리 사용되는 선택 사항으로, 더 복잡한 쿼리 요구 사항이 있을 수 있으며 MySQL 및 PostgreSQL의 관계형 데이터 모델링 기능을 활용할 수 있습니다. Amazon ElastiCache는 고성능 리더보드, pub/sub 메시징을 구축하고 자주 액세스하는 데이터를 캐싱하여 데이터베이스의 지연 시간과 로드를 줄이는 데 유용합니다. Amazon DynamoDB는 예측할 수 없는 액세스 패턴과 플레이어 및 게임 상태 데이터, 세션 데이터, 인벤토리 및 항목 스토어 또는 오버헤드가 최소화된 글로벌 데이터베이스를 원하는 사용 사례와 같은 사용 사례에 대해 거의 무제한의 처리량으로 확장할 수 있는 기능에 이상적인 완전관리형 NoSQL 데이터 스토어입니다.
- 리더보드 업데이트 또는 친구 요청 전송과 같이 백그라운드에서 수행할 수 있는 작업을 수행하려면 비동기 처리 워크플로를 사용해야 합니다. 이러한 유형의 작업을 Amazon SQS 대기열로 푸시하여 게임이 성장함에 따라 확장하거나 Amazon SNS 주제를 사용하여 병렬 처리를 위해 많은 소비자 애플리케이션 대기열 간에 작업을 분산하도록 게임 백엔드를 구성합니다. Lambda 함수 를 사용하여 AWS이벤트 기반 방식으로 처리를 수행하여 컴퓨팅 인프라 비용과 관리 오버헤드를 줄입니다. 수명이 길거나 여러 단계로 작업을 조정해야 하는 워크플로의 경우를 사용하여 전체 워크플로를 오케스트레이션하는 것이 좋습니다AWS Step Functions. Amazon EventBridge를 사용하여 AWS 서비스 및 사용자 지정 애플리케이션 이벤트에 응답하는 함수를 시작할 수 있습니다.

서버리스 기반 게임 백엔드 아키텍처

많은 게임 개발자는 인프라를 관리하는 대신 소프트웨어에 집중할 수 있는 기술을 사용하여 게임을 빌드하는 것을 선호합니다. 이 시나리오에서는 서버리스 아키텍처를 사용하는 것이 좋습니다.이 아키텍처를 사용하면 운영 오버헤드를 줄이면서 기능을 더 빠르게 빌드하고 릴리스할 수 있기 때문입니다. 서버리스 아키텍처는 서버를 설정, 관리 및 확장할 필요 없이 수요에 따라 동적으로 확장할 수 있는 클라우드 서비스를 사용하여 설계되었습니다. 다음 참조 아키텍처는 서버리스 아키텍처를 사용하여 게임을 빌드하는 방법을 보여줍니다.


AWS Cloud
VPC
Public subnet
NAT Gateway
Private subnet
Lambda Security group
Multiplayer - Live Scoreboard Functions
CacheGame
JoinGame
SendQuestion
ReceiveAnswer
StartGame
EndGame
Redis Security group
CACHE
Redis Real-time Game and Scoreboard
IoT Core
Vue.js Application
WebSockets
HTTP
Cognito
Multiplayer - Casual and Competitive Functions
OnConnect
LiveAdmin
LivePlayer
OnDisconnect
Single Player Game Functions
ActiveGamesList
GamePlay
GameAnswer
Leaderboard Get
Content Creation, Administration, and Marketplace Functions
GameActivate
GameHeaderPut
QuestionPut
GameGet
ListMarketplace
GetMarketplace
ListMyGames
PlayerGet
PlayerPut
ProgressionGet
PlayerWalletAdd
PlayerWalletGet
PurchaseGame
Analytics
Kinesis
Kinesis Data Analytics
Athena
S3
DynamoDB Tables
Connections
Player Profiles
Games
Player Progress
Game Detail
Inventory
Marketplace
Player Wallet
ActiveGames
Leaderboard
Game Connections
Shared Services
PlayerWallet Topic
MoneyAdjust
Leaderboard Topic
ScorePut
PlayerProgress Topic
PlayerProgress Put

서버리스 기반 게임 백엔드 참조 아키텍처

이 참조 아키텍처는 단일 플레이어 및 멀티플레이어 기능을 제공하는 웹 기반 트리비아 게임을 보여줍니다.

- 플레이어 인증: 플레이어는 플레이어 자격 증명 관리를 위해 사용자 디렉터리로 보안 인증을 제공하는 Amazon Cognito를 사용하여 인증합니다.
- 서버리스 함수로서의 게임 로직: 게임 기능 및 백엔드 비즈니스 로직은 이벤트에 대한 응답으로 시작되는 함수로 실행되므로 AWS Lambda 함수가 실행될 때만 비용을 지불하기 때문에 비용이 절감됩니다. Lambda는 원하는 프로그래밍 언어를 사용하여 각 게임 기능을 별도의 마이크로서비스로 작성할 수 있는 유연성을 제공합니다. 예를 들어 C#을 사용하여 Unity 게임을 빌드한 경험이 있는 경우 .NET Lambda 함수를 개발하도록 선택하거나 JavaScript로 웹 기반 게임의 프런트엔드와 백엔드를 프로그래밍하려는 경우 Node.js Lambda 함수를 개발하도록 선택할 수 있습니다.
- 게임 및 플레이어 데이터를 위한 NoSQL 데이터 스토어: DynamoDB를 사용하여 플레이어 및 게임 데이터를 저장합니다. 마이크로서비스에서 대량의 데이터를 저장하기 위해 특별히 구축되었습니다. 이 아키텍처에 설명된 대로 각 게임 기능의 데이터 스토리지 요구 사항에 대해 별도의 데이터 스토어를 사용하는 것이 모범 사례이므로 기능을 독립적으로 모니터링하고 관리할 수 있습니다. 또한 팀 내에서 기능 또는 서비스 소유권이 변경되면 분리 경계가 생성됩니다. 이 참조 아키텍처에서 DynamoDB 테이블은 연결 상태, 게임 세부 정보, 플레이어 진행 상황 및 리더보드 정보와 같은 데이터를 저장하는 데 사용됩니다.
- 단일 플레이어 게임플레이: 단일 플레이어 기능을 통해 플레이어는 게임 선택 및 재생, 리더보드 보기와 같은 작업을 수행할 수 있습니다. 이러한 기능은 적절한 Lambda 함수를 호출하여 DynamoDB 테이블에서 데이터를 가져오고 설정하는 Amazon API Gateway HTTP API로 호스팅되는 RESTful 백엔드 서비스로 구현됩니다. 게임 플레이가 완료되면 백엔드는 Lambda 함수를 비동기적으로 시작하는 Amazon SNS 주제에도 알림을 보내 플레이어의 진행 상황과 통계를 저장합니다.
- 멀티플레이어 게임플레이: 멀티플레이어 게임 기능을 사용하려면 플레이어가 point-to-point 통신을 위해 게임과 상호 작용하고 연결된 다른 플레이어로부터 업데이트를 브로드캐스트하고 수신할 수 있어야 합니다. WebSockets 구현은 퀴즈와 같은 경량 게임에서 point-to-point 통신에 적합합니다. 플레이어는 Amazon API Gateway WebSockets에 대한 WebSockets 연결을 설정할 수 있습니다.이 연결은 연결을 관리하고 플레이어에 대해 전송하거나 수신할 메시지가 있는 경우에만 Lambda 함수를 호출합니다. 플레이어 간에 one-to-many 통신이 필요한 사용 사례의 경우는 클라이언트가 주제를 구독하고 수신하는 메시지에 대해 조치를 취할 수 있도록 MQTT를 통해 WebSockets를 사용한 메시징에 대한 지원을 AWS IoT Core 제공합니다. 이 아키텍처에서 MQTT를 통한 WebSockets는 게임 내 라이브 업데이트 브로드캐스트 및 연결된 플레이어에게 질문과 같은 사용 사례를 지원하는 데 사용됩니다. 대신 메시지 전송을 위해 Redis Pub/Sub를 선택하거나 메시지 보존이 필요한 경우 Redis 스트림을 AWS IoT선택할 수 있습니다.

- VPC 지원 Lambda 함수를 사용하여 프라이빗 서브넷의 리소스에 액세스: 라이브 리더보드와 같이 지연 시간이 짧은 데이터 세트의 쿼리 시간을 줄이는 데 사용되는 Amazon ElastiCache와 같은 VPC의 프라이빗 서브넷에 있는 리소스에 액세스하도록 VPC 지원 Lambda 함수를 구성합니다.

자세한 내용은 사용자 지정 게임 백엔드 호스팅 지침을 참조하세요 AWS.

클라우드 게임 개발(CGD)

클라우드 게임 개발(CGD)은 게임 개발 수명 주기가 게임을 빌드, 테스트 및 개발하는 데 필요한 인프라와 도구를 말합니다. 게임 개발은 사용자 간에 협업하며 인프라 요구 사항은 개발 단계에서 자주 변경됩니다.

많은 게임 개발자가 전 세계에 분산된 원격 개발 팀을 수용하고 있으며, 이러한 유형의 개발을 지원하는 기술이 필요합니다. 게임 개발자는에서 이러한 환경의 전체 또는 일부를 호스팅 AWS 하고의 AWS 리전 글로벌 가용성을 사용하여 필요에 따라 컴퓨팅 및 스토리지를 확장하여 리소스를 사용자에게 더 가깝게 배치하고 개발 환경을 보다 비용 효율적으로 관리할 수 있습니다.

환경은 게임 개발자 요구 사항에 따라 다를 수 있지만 일반적으로 아티스트, 디자이너, 엔지니어, QA 테스터, 계약업체 및 기타 직원이 작업을 수행할 수 있는 개발자 워크스테이션이 포함됩니다. 이러한 환경에는 일반적으로 사용자가 변경 사항을 체크인할 수 있는 소스 코드 리포지토리로 구성된 빌드 팜과 개발된 아티팩트를 빌드, 패키징 및 테스트하기 위한 CI/CD 인프라도 포함됩니다.

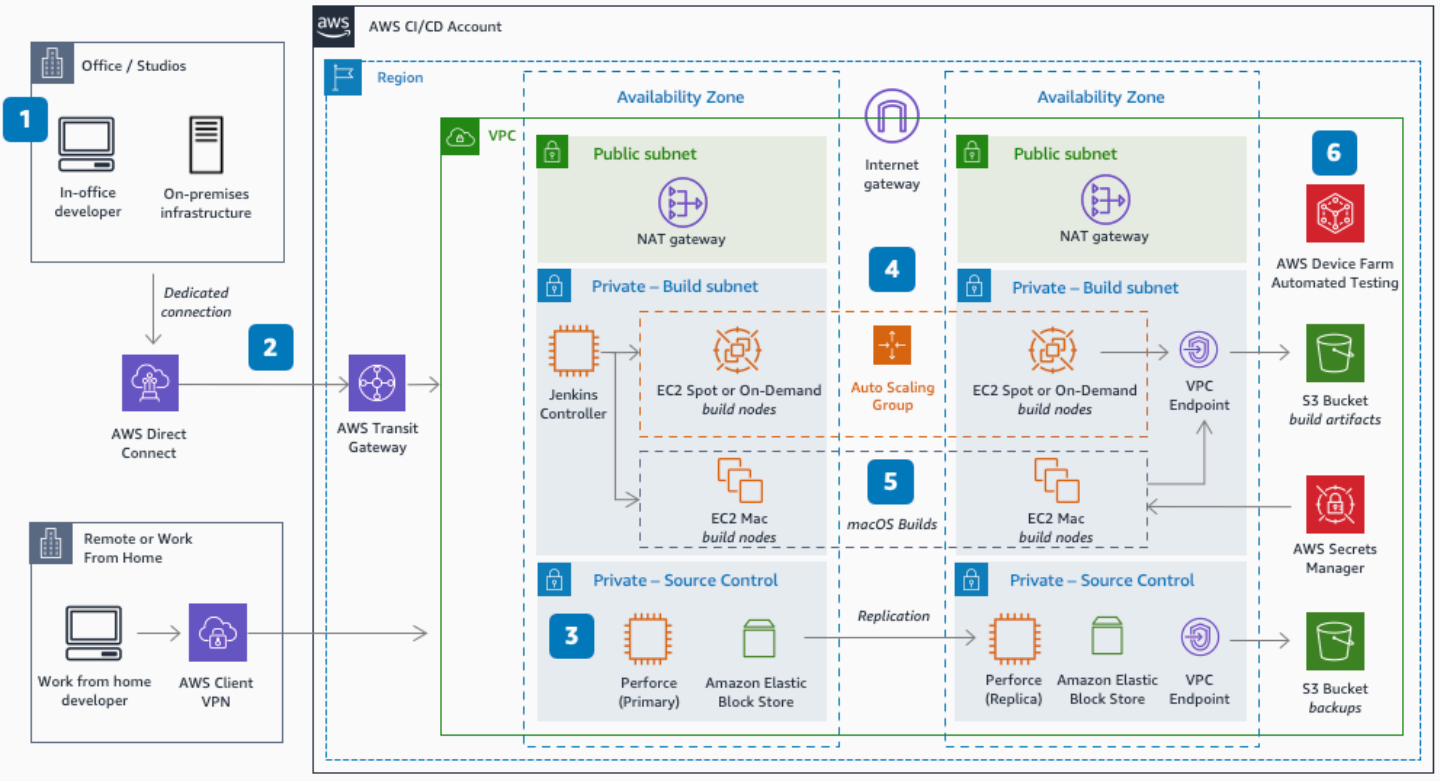
이러한 게임 프로덕션 아키텍처의 특징은 다음과 같습니다.

- 사용자는 웹 브라우저 또는 Amazon DCV와 같은 로컬 데스크톱 클라이언트를 통해 가상 워크스테이션에 액세스할 수 있어야 합니다.이 클라이언트는 사무실 또는 개발 스튜디오의 시스템에서 작업할 때 액세스할 수 있는 것과 동일한 소프트웨어 및 도구에 액세스할 수 있는 짧은 지연 시간 스트리밍 세션을 제공합니다. 일반적으로 클라우드 기반 서버인 이러한 가상 워크스테이션은 사용자가 LAN 또는 WAN을 통해 클라우드 환경에서 전적으로 프로젝트에 대해 협업하고 작업할 수 있도록 허용해야 합니다. 사용자가 시스템을 적극적으로 사용하지 않는 경우 소스 제어 리포지토리 또는 Amazon Elastic File System(EFS) 및 Amazon FSx와 같은 파일 시스템과 같은 내구성이 뛰어난 클라우드 스토리지에 작업을 백업해야 하며, 비용을 절감하려면 시스템을 종료해야 합니다.
- Perforce와 같은 소스 제어 리포지토리는 가용 영역 간 또는 Amazon S3와 같은 클라우드 스토리지에 백업이 저장된 온프레미스 환경 간 복제를 사용하여 고가용성으로 설계되어야 합니다. 예를 들어 클라우드 기반 Perforce 서버에는 동일한 리전의 다른 가용 영역에서 호스팅되는 대기 서버로 복제하는 한 가용 영역에서 호스팅되는 기본 커밋 서버가 포함되어야 합니다.

- 게임 개발 빌드 팜 리소스는 필요에 따라 컴퓨팅 리소스가 프로비저닝되도록 자동 조정으로 설계되어야 하며, EC2 스팟 인스턴스를 사용하여 빌드에 필요한 서버 수를 확장할 때 발생하는 비용을 줄여야 합니다.

클라우드 게임 개발: CI/CD

CI/CD 인프라는 팀 크기에 관계없이 게임을 개발하여 반복 시간을 개선하고, 신뢰성을 구축하고, 효율적인 배포를 구축하고, 개발 및 릴리스 프로세스를 더 잘 제어하여 플레이어에게 고품질 게임 경험을 제공할 때 중요합니다. 게임 개발 CI/CD 파이프라인은 일반적으로 가용성이 높은 소스 제어 서버 및 스토리지, 빌드를 실행하는 컴퓨팅 리소스, 자동 테스트를 수행하는 소프트웨어, 개발 머신의 적절한 네트워크 연결로 구성됩니다. 다음 참조 아키텍처는 원격 또는 온프레미스 게임 개발 환경에서 로 게임 빌드를 오프로드 AWS 클라우드 하여 개발자가 새 빌드 팜을 마이그레이션하거나 빌드하는 데 도움이 되는 방법을 보여줍니다.



게임 빌드를 클라우드로 오프로드

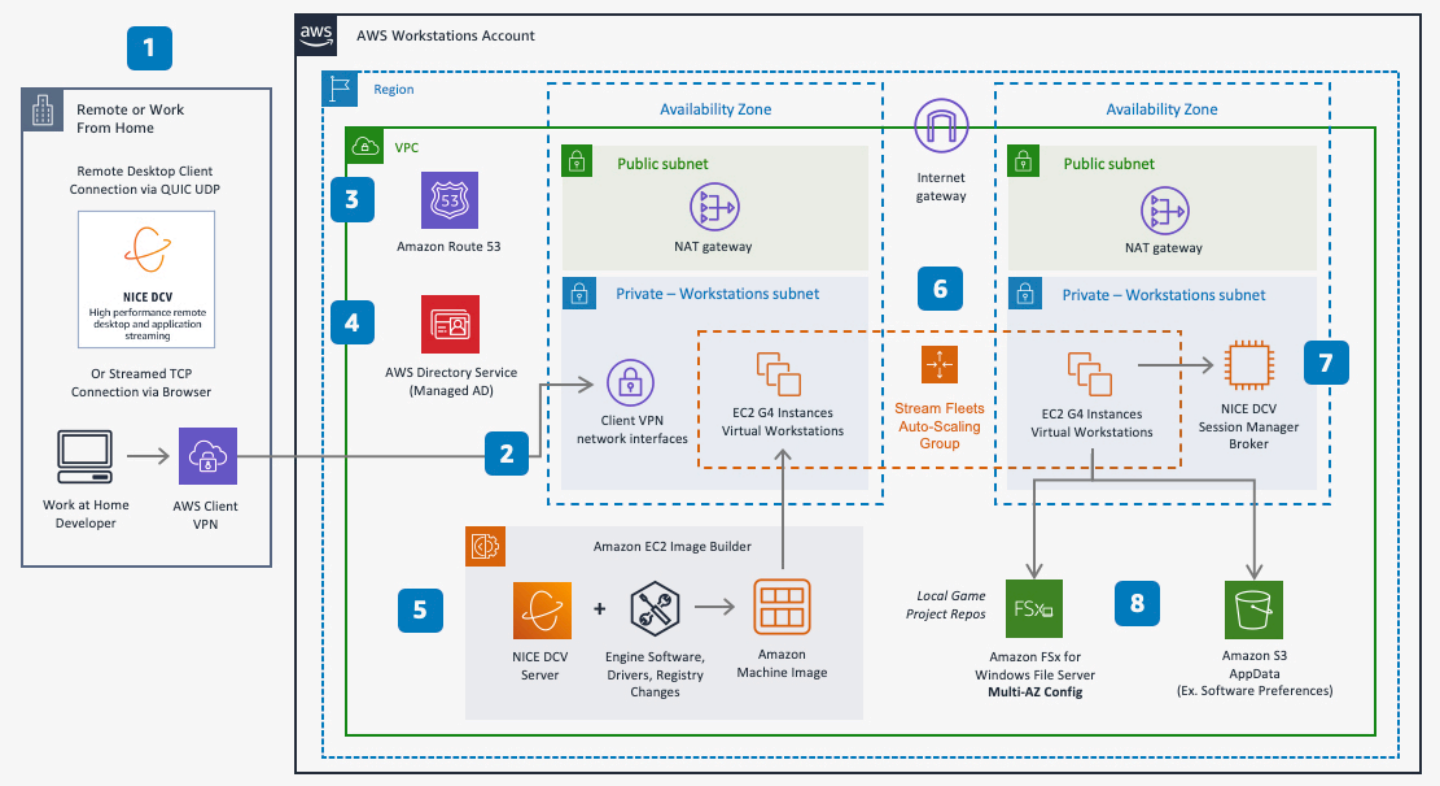
1. AWS Direct Connect 는 사무실 내 개발자를 AWS 위해 지연 시간이 짧은 프라이빗 전용 연결을 제공합니다. 원격 개발자는와 같은 제로 트러스트 기술 AWS Verified Access또는 AWS Client VPN과 같은 가상 프라이빗 네트워크(VPN)를 사용합니다.
2. AWS Transit Gateway 는 VPCs.

3. Perforce는 Amazon EBS 스토리지가 지원하는 소스 및 버전 제어(CI)를 관리하여 빠르게 액세스할 수 있는 영구 데이터를 제공합니다. Perforce Helix Core는에서 사용할 수 있습니다AWS Marketplace.
4. 개발자가 브랜치에 연결된 Perforce에 변경 사항을 푸시할 때 Jenkins에서 빌드(CD)를 시작합니다. Perforce는 Jenkins에 대한 JSON 페이로드 POST를 시작합니다. Jenkins 컨트롤러는 엔진 헤드리스 CLI 명령을 호출하여 임시, Docker 노드(예: Amazon EC2 스팟 인스턴스 또는 Amazon EC2 온디맨드 인스턴스)에서 빌드 프로세스를 실행하고 병렬화합니다. 개발자는 로드 밸런서 뒤의 각 가용 영역에 하나씩 있는 두 개의 Jenkins 컨트롤러를 사용하여 가용성을 높일 수 있습니다. 일부 게임 엔진의 경우 개발자는 동시 빌드가 실행될 때마다 빌드 컨텍스트에 대한 라이선스를 판매하기 위해 추가 서브넷에 구성된 추가 라이선스 인프라가 필요할 수 있습니다.
5. iOS 빌드의 Xcode 부분은 .IPA 파일에 서명, 빌드 및 내보내기 위해 Amazon EC2 Mac 인스턴스로 오프로드되어 프로세스를 분할하고 빌드 시간을 줄입니다. 에는 프로비저닝 프로필, 프라이빗 키 및 인증서가AWS Secrets Manager 있습니다.
6. 빌드 아티팩트는 Amazon S3로 전송되어 성공 또는 실패 알림을 보냅니다. 모바일 디바이스에 대한 자동 테스트를 AWS Device Farm 활성화합니다.

클라우드 게임 개발: 워크스테이션

게임 개발에는 종종 다양한 위치에서 원격으로 작업하는 분산 팀이 포함되며, 공유 인프라에 대한 액세스와 지리적으로 분산된 공동 개발을 지원하는 기능이 필요합니다. work-for-hire 스튜디오 및 원격 작업 사용을 포함하여 보다 분산된 게임 개발 프로세스에 대한 이러한 추세로 인해 생산성, 공유 전문 지식 및 민첩한 개발 관행을 촉진하기 위해 강력한 기술과 워크플로를 구현해야 합니다.

다음 참조 아키텍처는 Amazon DCV 프로토콜을 사용하여 원격 게임 개발 워크스테이션을 호스팅하는 데를 사용하는 AWS 방법을 보여줍니다.



Amazon DCV를 사용하여 어디서나 게임 개발 스트리밍

1. Amazon DCV는 4K, 60-FPS 스트리밍을 지원하는 스트리밍 프로토콜입니다. 브라우저를 사용하는 개발자는 TCP 연결을 통해 연결하는 반면 데스크톱 클라이언트는 성능 향상을 위해 포트 8443을 통해 QUIC UDP를 사용할 수 있습니다.
2. 개발자는 소스 네트워크 주소 변환(SNAT)을 사용하여 워크스테이션 서브넷의 네트워크 인터페이스에 안전하게 연결하기 위해 AWS Client VPN을 사용합니다.
3. Amazon Route 53는 VPC의 리소스에 대한 프라이빗 DNS와 인바운드 및 아웃바운드 DNS 전달을 제공합니다.
4. Directory Service 는 관리형 Microsoft Active Directory를 제공하여 개별 사용자에게 매핑된 로컬 게임 프로젝트 스토리지를 활성화합니다.
5. 워크스테이션은 Image Builder로 빌드된 Amazon Machine Image(AMI)를 사용하여 생성됩니다. 이 미지에는 Amazon DCV 서버, 개발자 소프트웨어, 레지스트리 변경 사항, NVIDIA 게임 드라이버 또는 주변 드라이버와 같은 드라이버가 포함됩니다. 워크스테이션에 사용되는 일반적인 AMIs가 AWS Marketplace 포함됩니다.
6. 워크스테이션 플릿은 GPUs를 제공하는 그래픽 Amazon EC2 인스턴스 유형을 사용하며 EC2 Auto Scaling 그룹을 사용하여 확장됩니다.
7. Amazon DCV 세션 관리자 브로커를 사용하면 Amazon DCV 세션을 관리할 수 있습니다.

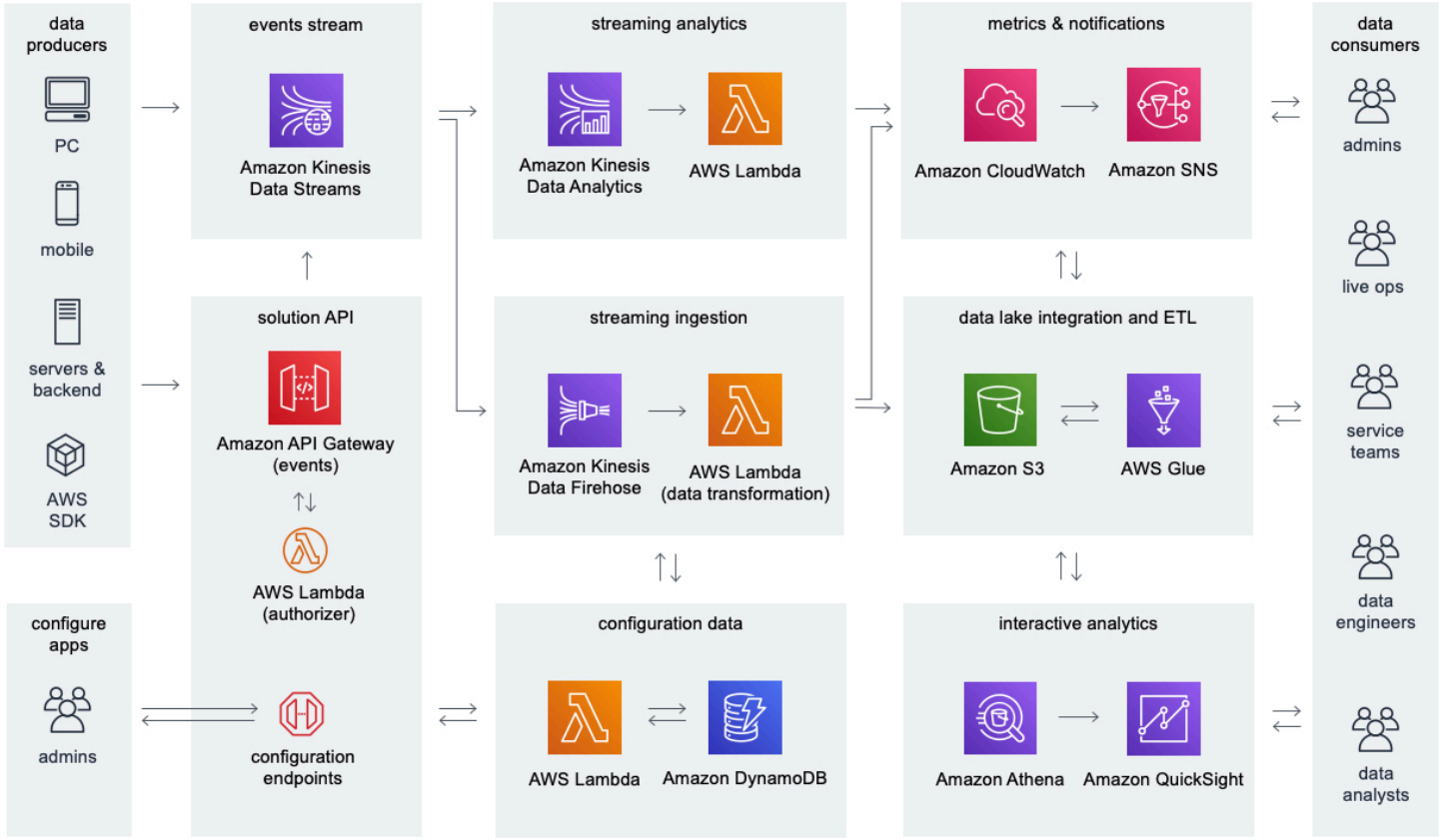
8. 프로젝트의 로컬 파일 스토리지는 Amazon FSx for Windows File Server에서 호스팅됩니다. 개발자는 로컬 스토리지에서 소스 제어로 푸시하여 별도의 CI/CD 파이프라인에 커밋합니다.

게임 분석 파이프라인

게임 개발자는 게임 플레이 경험을 개선하여 플레이어 기반을 유지하고 성장시킬 수 있도록 플레이어 행동을 더 잘 이해할 수 있는 방법을 점점 더 찾고 있습니다. 게임 분석은 게임 및 관련 서비스에서 생성된 데이터를 이해하고 분석하는 데 필요한 기술 인프라 및 프로세스를 나타냅니다. 이를 위해서는 일반적으로 Game Analytics Pipeline 솔루션 구현과 같은이 end-to-end 프로세스를 지원할 수 있는 분석 파이프라인 아키텍처를 사용해야 합니다. https://aws.amazon.com/solutions/implementations/game-analytics-pipeline/

게임 분석 아키텍처의 특징은 다음과 같습니다.

- 데이터 소스는 JSON과 같은 일반적인 형식으로 데이터를 전송하며, 일반적으로 게임 서버 및 게임 백엔드 서비스와 PC, 모바일 디바이스, 게임 콘솔을 포함한 게임 클라이언트를 포함합니다.
- 게임 분석 파이프라인은 원시 데이터를 수집 및 저장하고 사용 가능한 출력 형식으로 처리하는 전체 워크플로를 자동화하여 최종 사용자 및 분석 애플리케이션과 같은 데이터 소비자가 효율적이고 비용 효율적으로 분석할 수 있도록 합니다.
- 게임 분석 파이프라인은 게임이 성장함에 따라 확장할 대량의 실시간 데이터를 수집하고 처리할 수 있도록 지원합니다.
- 실시간 및 배치 보고 사용 사례를 모두 지원합니다. 예를 들어 실시간 대시보드 및 알림은 일반적으로 라이브 운영 팀에서 게임 인프라와 플레이어 동작을 모니터링하여 문제를 감지하는 데 사용됩니다. 데이터 분석가 팀은 일반적으로 필요에 따라 배치 보고를 사용하여 시간 경과에 따른 추세를 파악합니다.



게임플레이 원격 측정을 위한 서버리스 게임 분석 파이프라인

게임 데이터는 게임 클라이언트, 게임 서버 및 기타 애플리케이션에서 수집됩니다. 스트리밍 데이터는 데이터 레이크 통합 및 대화형 분석을 위해 Amazon S3에 수집됩니다. 스트리밍 분석은 실시간 이벤트를 처리하고 지표를 생성합니다. 데이터 소비자는 Amazon CloudWatch의 지표 데이터와 Amazon S3의 원시 이벤트를 분석합니다.

- 솔루션 API 및 구성 데이터: Amazon API Gateway를 사용하여 게임 분석 파이프라인을 관리하고 Lambda 함수를 사용하여 Amazon DynamoDB에 구성 데이터를 저장하기 위한 REST API를 제공합니다. 이 API 또는 관리를 위한 사용자 지정 명령줄 인터페이스를 기반으로 내부 포털을 구축할 수 있습니다. 또한 REST API는 데이터 소스에서 게임 플레이 데이터를 수집하고 실시간 처리 및 스토리지로의 수집을 위해 원격 측정 데이터를 Amazon Kinesis Data Streams에 전달하기 위한 서버 인증을 제공합니다.
- 이벤트 스트림: Amazon Kinesis Data Streams는 게임에서 스트리밍 데이터를 캡처하고 Amazon Data Firehose 및 Amazon Managed Service for Apache Flink에서 실시간 데이터 처리를 허용합니다.

- 스트리밍 분석: Managed Service for Apache Flink는 Kinesis Data Streams의 스트리밍 이벤트 데이터를 분석하고 Lambda 함수를 사용하여 CloudWatch에 게시되는 사용자 지정 지표 및 알림을 생성할 수 있습니다.
- 지표 및 알림: Amazon CloudWatch를 사용하여 솔루션의 지표, 로그 및 경보를 모니터링합니다. Amazon SNS를 사용하여 대기 중인 엔지니어 및 기타 데이터 소비자에게 알림을 보냅니다.
- 스트리밍 수집: Firehose를 사용하여 Kinesis Data Streams에서 스트리밍 데이터를 사용하고 Amazon S3의 데이터 레이크로 전송하여 장기 저장, 변환 및 다른 데이터와의 통합을 수행합니다.
- 데이터 레이크 통합 및 ETL: ETL 처리 워크플로 AWS Glue 에를 사용하고 AWS AWS Glue Data Catalog에서 메타데이터를 구성하여 유연한 분석 도구와 통합하기 위한 데이터 레이크의 기반을 제공합니다.
- 대화형 분석: 최종 사용자는 Amazon Athena를 사용하여 Amazon S3에 저장된 데이터 세트에 대해 임시 대화형 쿼리를 수행할 수 있으며 Quick Suite를 사용하여 대시보드를 구축할 수 있습니다.

를 사용하여 계정에 배포할 수 있는 분석 파이프라인의 자동 참조 구현은 Game Analytics 파이프라인을 참조하세요 AWS CloudFormation.

정의

AWS Well-Architected Framework는 운영 우수성, 보안, 신뢰성, 성능 효율성, 비용 최적화 및 지속 가능성이라는 6가지 원칙을 기반으로 합니다. AWS 는 게임 워크로드를 위한 state-of-the-art 아키텍처를 설계할 수 있는 여러 핵심 구성 요소를 제공합니다. 이 섹션에서는 키 정의에 대한 개요를 제공합니다.

이 백서의 목적상 게임 아키텍처는 게임을 구축하고 운영하는 데 필요한 백엔드 기술 인프라를 포함합니다. 일부 게임에는 소셜, 멀티플레이어 또는 기타 온라인 기능이 없을 수 있으며이 백엔드 기술 인프라의 특정 측면을 사용할 필요가 없을 수 있습니다. 게임 아키텍처를 지원하기 위해 자주 배포되는 다양한 유형의 워크로드에 대한 자세한 내용은 시나리오를 참조하세요.

AWS 클라우드 인프라는 리전 및 가용 영역을 중심으로 구축됩니다.

- 리전은 가용 영역이 여러 개인 전 세계의 물리적 위치입니다.
- 가용 영역은 각각 중복 전원, 네트워킹 및 연결을 갖춘 하나 이상의 개별 데이터 센터로 구성되며 별도의 시설에 보관됩니다.

게임의 특성에 따라 플레이어의 성능 향상과 같은 이유로 게임 아키텍처의 특정 구성 요소를 여러 리전에 배포하거나 위치에 따라 플레이어에게 사용자 지정 경험을 제공할 수 있습니다.

다양한 유형의 게임이 있으며 게임을 지원하는 데 필요한 백엔드 기술 인프라는 개발 중인 게임 유형에 따라 다릅니다. 예를 들어 인기 있는 유형의 게임에는 1인칭 슈팅 게임(FPS), 역할 플레이 게임(RPG), 대규모 멀티플레이어 온라인 게임(MMOG), 배틀 로일(BR), 스포츠 게임, 퍼즐 게임 등이 포함될 수 있습니다. 또한 성능 특성이 서로 다른 턴 기반 및 동시 재생과 같이 게임의 아키텍처에 영향을 미치는 다양한 게임 상호 작용 모드가 있습니다.

게임은 데스크톱, 웹, 모바일, 콘솔, 증강 현실(AR), 가상 현실(VR), 게임 스트리밍 솔루션과 같은 최신 상호 작용 모드를 포함한 하나 이상의 게임 시스템에서 재생되도록 개발되었습니다. 게임은 일반적으로 교차 시스템 게임 플레이를 지원합니다. 즉, 플레이어는 게임 진행 상황을 저장하고 다른 시스템에서 게임 플레이를 재개할 수 있을 뿐만 아니라 다른 시스템의 플레이어와 게임 플레이 세션을 시작할 수 있습니다.

비디오 게임 수익화를 통해 게임 게시자는 광고, 디지털 및 소매 기반 게임 구매, 마이크로트랜잭션이라고 하는 다운로드 가능한 콘텐츠(DLC)의 게임 내 구매, 게임 플레이에 필요한 유료 구독과 같은 다양한 전략을 사용하여 수익을 창출할 수 있습니다. 게임 업계에서 가장 일반적인 핵심 성과 지표(KPIs)는 다음과 같습니다.

- 일일 활성 사용자(DAU)

- 월간 활성 사용자(MAU)
- 동시 사용자(CCU)
- 세션 지속 시간
- 설치당 비용(CPI)
- 플레이어 수명 값(LTV)
- 사용자당 평균 수익(ARPU)

게임 시스템

비디오 게임은 클라이언트 입력 제어, 그래픽, 클라이언트 소프트웨어(게임 클라이언트라고 함) 및 하드웨어, 경우에 따라 게임 플레이를 지원하는 시스템 전용 기능을 제공하는 게임 시스템에서 재생되도록 개발되었습니다.

게임 시스템은 일반적으로 다음 범주로 구분됩니다.

- 콘솔: Sony PlayStation, Microsoft Xbox, Nintendo Switch와 같은 인기 있는 예제를 포함하여 게임 플레이용으로 설계된 목적별 엔터테인먼트 시스템입니다. 콘솔은 게임 시스템 공급자가 제조한 콘솔 하드웨어에 물리적 또는 디지털 방식으로 배포된 게임 콘텐츠를 설치하여 게임을 플레이할 수 있는 기능을 제공합니다. 이 정의에서 콘솔은 휴대용 또는 고정형일 수 있으며 홈 엔터테인먼트 시나리오에서 사용하기 위한 것입니다.
- PC(개인용 컴퓨터): 플레이어가 사용자 지정할 수 있는 클라이언트 머신에 설치된 컴퓨터 소프트웨어를 사용하여 재생되는 게임입니다. 이러한 이유로 PC 게임은 플레이어에게 인기가 높습니다. PC 게임이 제공하는 유연성과 제어 기능 때문입니다.
- 웹: 웹 브라우저를 사용하여 재생하도록 설계되고 일반적으로 플레이어가 운영 체제에 관계없이 게임에 액세스할 수 있는 이점을 제공하는 게임입니다.
- 모바일: 일반적으로 스마트폰 운영 체제인 휴대폰에서 재생하도록 개발된 게임입니다. 모바일 게임은 일반적으로 디지털 앱 스토어에서 다운로드되어 휴대폰에 설치됩니다.

앞서 언급한 시스템 외에도 여전히 비교적 새롭고 성장하며 더 지배적인 시스템에 비해 시장 점유율이 훨씬 작은 초기 시스템도 있습니다. 이 범주의 게임 시스템의 예로는 AR, VR 및 게임 스트리밍이 있으며, 이를 클라우드 게임이라고도 합니다.

게임 스트리밍에는 클라우드에서 게임 플레이를 렌더링하고 일반적으로 브라우저인 씬 클라이언트로 스트리밍하는 작업이 포함됩니다. 게임 스트리밍을 통해 플레이어는 일반적으로 게임 스트리밍 서비스 공급자가 클라우드에서 원격으로 호스팅하는 게임을 플레이할 수 있습니다. 게임 스트리밍에서 플

레이어는 브라우저 또는 클라우드 게임 서비스 공급자(게임 시스템)가 제공하는 씬 클라이언트를 통해 클라우드 기반 게임에 연결합니다.

게임 서버

게임 서버는 게임에 대한 컴퓨팅 인프라의 가장 중요한 측면 중 하나를 나타냅니다. 전용 게임 서버라고도 하는 게임 서버는 멀티플레이어 게임을 개발할 때 또는 게임 플레이 이벤트의 서버 권한 처리가 필요할 때 사용됩니다. 게임 서버는 게임 아키텍처의 중심에 있으며, 플레이어 및 게임 상태 관리, 연결된 게임 클라이언트와 게임 서버 간의 상호 작용 관리 등 코어 로직이 실행되는 위치 역할을 합니다. 게임 서버는 플레이어의 게임 클라이언트에서 입력을 처리하고 연결된 다른 플레이어에게 실시간으로 적절하게 배포하는 역할을 하므로 일반적으로 게임 아키텍처에서 성능에 가장 민감한 측면 중 하나입니다. 성능이 좋지 않은 게임 서버는 게임 경험의 전반적인 성능에 영향을 미칩니다. 따라서 게임 서버 성능을 최적화하고 특히 게임 시작 또는 피크 게임 플레이 기간에 충분한 용량을 제공해야 합니다.

이 문서의 목적상 게임 서버 또는 게임 서버 인스턴스는 하나 이상의 게임 서버 프로세스를 호스팅하는 가상 머신과 같은 컴퓨팅을 나타냅니다. 게임 서버 프로세스는 플레이어가 플레이어 세션을 통해 연결할 수 있는 실행 중인 게임의 인스턴스인 게임 세션을 호스팅하는 게임 서버 빌드의 단일 인스턴스를 나타냅니다. 이러한 이유로 게임 세션과 호스팅하는 게임 서버 프로세스 간의 묵시적인 일대일 관계로 인해 종종 게임 서버 프로세스 또는 게임 세션을 상호 교환적으로 참조합니다. 에는 컴퓨팅이 게임 서버를 호스팅할 수 있는 AWS여러 옵션이 있으며, 이를 통해 리소스의 탄력적 프로비저닝을 통해 확장 가능한 클라우드 기반 용량에 액세스할 수 있습니다.

Amazon EC2는 여러 버전의 Linux 및 Windows를 지원하는 인스턴스라고 하는 클라우드 기반 가상 서버를 제공합니다. 인스턴스를 생성하고 다른 서버 또는 가상 머신처럼 직접 관리할 수 있습니다. 일반적으로 효율성을 개선하고 비용을 절감하기 위해 여러 게임 서버 프로세스가 인스턴스에 배포됩니다. Amazon EC2는 컴퓨팅 인프라를 가장 잘 제어하려는 게임 서버에 적합합니다.

Amazon GameLift는 클라우드에서 전용 게임 서버 호스팅을 위한 완전관리형 솔루션과 GameLift FlexMatch를 사용한 매치메이킹과 같은 추가 기능을 제공합니다. GameLift는 Amazon EC2를 기반으로 추상화 계층을 제공하여 게임 서버 관리를 간소화하고 대부분의에서 사용할 수 AWS 리전 있으므로 플레이어와 가까운 게임 서버를 호스팅하여 지연 시간을 줄이고 고가용성을 달성하며 스팟 인스턴스를 사용하여 비용을 크게 절감할 수 있습니다. GameLift는 기존 게임 백엔드에 통합할 수 있지만 자체 게임 서버 관리 및 매치메이킹 솔루션을 개발하지 않고에서 관리하고 게임이 성장함에 따라 확장할 AWS 수 있는 솔루션을 선호하는 게임 개발자에게 특히 유용합니다.

Amazon Elastic Container Service(Amazon ECS)는 Docker 기반 컨테이너를 실행하는 데 사용할 수 있는 완전 관리형 컨테이너 오케스트레이션 서비스입니다. Amazon Elastic Kubernetes Service(Amazon EKS)를 사용하여 Kubernetes를 사용하여 빌드된 Docker 기반 컨테이너를 실행할 수

도 있습니다. Amazon ECS 및 Amazon EKS에서 제공하는 것과 같은 컨테이너 기술을 사용하면 많은 게임 서버 프로세스 또는 기타 게임 애플리케이션 인스턴스를 EC2 인스턴스에 효율적으로 패킹하여 컴퓨팅 사용률을 개선할 수 있습니다.

컨테이너를 사용하면 개발 중에 로컬 시스템에서 개발자가 사용하는 것과 동일한 Docker 이미지 운영 런타임을 사용하여 애플리케이션을 호스팅하여 개발자 생산성을 높일 수도 있습니다. 컨테이너를 실행하기 위한 서버리스 컴퓨팅 솔루션이며 Amazon EKS 및 Amazon ECS와 호환되는 AWS Fargate를 사용하여 운영 오버헤드를 더욱 줄일 수 있습니다. Fargate는 컨테이너가 실행되는 기본 인스턴스를 운영할 책임 없이 컨테이너에서 게임 서버를 실행하려는 사용 사례에 가장 적합합니다.

AWS Outposts 를 사용하여 데이터 센터 또는 온프레미스 시설에서 AWS 인프라와 서비스를 실행할 수 있습니다. 이를 통해 게임이 온프레미스 환경에서 실행되고 AWS 동일한 인프라를 사용하여 하이브리드 클라우드 채택 전략을 지원할 수 있습니다. AWS 로컬 영역은 게임 서버 및 기타 지연 시간에 민감한 워크로드를 플레이어 또는 개발 팀과 더 가깝게 실행할 수 AWS 리전 있도록 하는의 확장 역할을 합니다. 또한 게임 서버의 글로벌 네트워크 지연 시간을 줄이기 위해 AWS Global Accelerator를 사용하여 게임 서버로의 플레이어 트래픽 성능을 개선할 수 있습니다.

AWS Lambda 는 서버를 프로비저닝하거나 관리하지 않고 코드를 실행하는 서버리스 컴퓨팅 서비스로, 턴 기반 게임이나 간단한 컴퓨팅 요구 사항, 작은 코드베이스가 있고 상태 비저장 마이크로서비스 아키텍처를 사용하여 게임 플레이 기능을 설계할 수 있는 게임 서버 사용 사례에 유용합니다. Lambda 함수는 장기 실행 게임 서버 프로세스를 실행하는 대신 이벤트 기반 요청별로 실행된다는 점에 유의해야 합니다. Lambda는 개발자가 코드를 호스팅하기 위해 선택할 수 있는 기본 애플리케이션을 즉시 사용할 수 있기 때문에이 백서의 옵션에 대한 가장 많은 런타임 추상화를 제공합니다.

게임 서버 호스팅에 대한 접근 방식을 선택할 때 운영 오버헤드, 레거시 코드베이스, 성능 요구 사항 및 규모를 비롯한 다양한 요구 사항을 고려하세요. EC2 인스턴스와 컨테이너는 클라우드로 이동하는 데 가장 작은 변경이 필요하므로 레거시 코드베이스에 적합한 옵션이며, EC2 인스턴스를 사용하여 컴퓨팅 인스턴스의 리소스를 전용으로 사용할 수 있고 컨테이너를 사용하면 관리 및 높은 사용률을 더 간단하게 달성할 수 있습니다. 서버리스 함수는 이벤트에 대한 응답으로만 실행되는 코드를 정의하는 데 사용할 수 있는 최고 수준의 추상화를 제공하므로 비용을 절감할 수 있습니다.

게임 클라이언트

게임 클라이언트는 플레이어가 게임을 플레이하는 데 사용하는 소프트웨어 및 하드웨어 디바이스를 나타냅니다. 게임 클라이언트는 플레이어의 입력을 처리를 위해 서버로 전송되는 메시지로 변환하는 소프트웨어를 제공하며, 서버에서 수신되는 응답을 처리하고 그래픽과 같은 출력을 플레이어에게 렌더링할 책임이 있습니다. 실시간 네트워크 멀티플레이어 게임에서 게임 클라이언트는 일반적으로 게임 플레이 세션 기간 동안 게임 서버에 대한 지속적인 네트워크 연결을 유지하여 네트워크 지연 시간을

줄이고 처리 시간을 최소화합니다. 그러나 게임 클라이언트는 REST를 사용하여 게임 서버 또는 백엔드 서비스와 상호 작용할 수도 있습니다.

메시징

게임에는 일반적으로 세 가지 주요 범주의 메시지가 있습니다.

- 게임 초대 또는 푸시 알림과 같은 특정 사용자 또는 사용자 집단을 대상으로 하는 플레이어 참여 메시징
- 게임 내 채팅과 같은 플레이어 간의 그룹 메시징
- 둘 이상의 애플리케이션을 통합하는 데 사용되는 JSON 메시지와 같은 Service-to-service 메시징

이러한 유형의 메시지를 보내고 받는 일반적인 전략은 게시자-구독자 및 비동기 처리 아키텍처 패턴을 사용하는 것입니다.는 게임에서 메시징을 구현하는 데 도움이 되는 여러 서비스를 AWS 제공합니다.

- Amazon Simple Notification Service(SNS): 게시자와 구독자 간에 pub/sub 아키텍처 패턴을 사용하여 메시지를 전송하는 관리형 서비스입니다. 게시자는 API를 사용하여 Amazon SNS는 구독 애플리케이션에 메시지를 비동기적으로 전송하고 가장 널리 사용되는 푸시 알림 서비스 중 일부를 지원하여 모바일 클라이언트 또는 데스크톱에 직접 푸시 알림을 전송할 수 있습니다. Amazon SNS는 클라이언트에 대한 푸시 알림과 service-to-service 메시징 사용 사례에 사용할 수 있습니다.
- Amazon Simple Queue Service(SQS): 각에서 사용되는 프로그래밍 언어에 관계없이 게임 서버와 게임을 쉽게 통합할 수 있는 완전 관리형 대기열 서비스입니다. 데이터베이스에서 리더보드 또는 재생 시간 값을 업데이트하는 등 백그라운드에서 많은 게임 작업을 분리하고 처리할 수 있습니다. 이 접근 방식은 게임의 다양한 부분을 분리하고 플레이어 대면 기능을 백엔드 처리에서 독립적으로 확장하는 효과적인 방법입니다.
- Amazon Managed Streaming for Apache Kafka(MSK): 널리 사용되는 오픈 소스 솔루션인 Apache Kafka를 사용하여 데이터 스트리밍 및 생산자 또는 소비자 애플리케이션 구축을 간소화하는 완전 관리형 서비스입니다. Kafka는 일반적으로 실시간 스트리밍 데이터를 수집하고 처리하는 데 사용되며 service-to-service 메시징에 사용할 수 있습니다.
- Amazon ElastiCache(Redis OSS): 채팅룸 애플리케이션 개발 및 고성능 service-to-service 메시징에 일반적으로 사용되는 Redis의 인기 있는 pub/sub 기능에 대한 지원을 포함하는 완전 관리형 인 메모리 데이터 스토어를 제공합니다. 또한 Redis는 목록 및 세트와 같은 풍부한 데이터 형식을 지원하므로 개발자는 고성능 대기열에 Redis를 사용할 수 있습니다.
- Amazon Pinpoint: 이메일, SMS, 음성 및 푸시 알림을 통해 사용자 참여 메시징을 제공합니다. 예를 들어 Amazon Pinpoint를 사용하여 플레이어에게 사용자 참여 메시지를 전달하여 게임에 다시 초대

하고 다중 인증 토큰, 주문 확인 및 암호 재설정 이메일 지원과 같은 트랜잭션 사용 사례에 사용할 수 있습니다.

라이브 게임 운영(라이브 운영)

라이브 게임 운영(라이브 운영)은 게임을 라이브 서비스로 취급하고 출시된 게임에 새로운 기능, 업데이트, 프로모션, 게임 내 이벤트 및 개선 사항을 지속적으로 제공하여 플레이어 커뮤니티의 경험을 개선하는 게임 관리 및 운영 스타일입니다.

전통적으로 게임은 서비스가 아닌 제품으로 제공되었으며, 새로운 콘텐츠와 기능은 출시된 제품이 아닌 후속 릴리스 또는 시퀀스에 자주 통합되었습니다. 게임 관리에 대한 Live Ops 접근 방식을 사용하면 게임 운영 팀이 실험, 프로모션, 게임 내 이벤트, 혁신을 통해 게임을 시작하고 참여 플레이어 커뮤니티를 유지하여 플레이어가 계속 즐거운 시간을 보낼 수 있습니다.

이 접근 방식은 새로운 플레이어 참여 전략을 잠금 해제하고 반복적인 수익 흐름을 제공한다는 이점이 있지만 더 많은 운영 전문 지식이 필요합니다. 예를 들어 성공적인 Live Ops 전략을 구현하려면 개발자가 클라우드 서비스와 통합하거나 자체 백엔드 기술 인프라를 운영해야 할 수 있습니다. 또한 플레이어 경험에 부정적인 영향을 미칠 수 있는 게임 또는 플레이어 커뮤니티 내에서 발생하는 문제를 식별하고 대응할 수 있는 효과적인 방법이 필요합니다.

운영 우수성

운영 우수성 원칙은 클라우드 기반 게임을 대규모로 배포하고 운영하는 모범 사례에 중점을 둡니다. 운영 우수성에 집중하여 긍정적인 플레이어 경험을 유지하고 예방 조치를 구현하여 경험에 영향을 미치는 문제에 대비하고 복구하는 것이 중요합니다.

중점 영역

- 설계 원칙
- 라이브 작업
- 계정 구조
- 게임 배포
- 상태 모니터링
- 로드 테스트.
- 시간 경과에 따른 최적화
- 리소스

설계 원칙

Well-Architected Framework 백서의 설계 원칙 외에도 다음 설계 원칙은 게임 구축 및 운영에서 운영 우수성을 달성하는 데 도움이 될 수 있습니다.

- 게임 운영 팀을 위한 측정 가능하고 달성 가능한 목표를 정의하고 필요에 따라 조정합니다. 게임의 히트 중심 특성으로 인해 게임이 시작될 때 게임을 플레이할 플레이어 수 또는 플레이어가 지속적인 게임 운영에 대해 기대하는 바를 미리 결정하기가 어렵습니다. 이해관계자와 함께 야심차지만 달성 가능한 목표를 설정하고 게임 개발 팀이 플레이어 경험을 최적화하는 동안 게임이 프로젝션을 초과하고 축소될 경우 확장할 수 있는 접근 방식을 설계하는 것이 중요합니다. 이러한 요구 사항을 충족하고 비즈니스 및 기술 이해관계자를 게임 운영의 대상 목표에 맞게 사전에 적절하게 준비하고 테스트합니다. 목표가 정의되면 게임 팀은 게임 백엔드 인프라를 계획, 설계, 프로비저닝, 테스트, 배포 및 운영하는 동안 비용과 성능 간의 적절한 균형을 달성할 수 있습니다.
- 운영 런북을 사용하여 게임 출시 및 특별 이벤트와 관련된 활동 규모 조정 계획: 게임 운영 팀은 비즈니스 이해관계자와 협력하여 이벤트에 대한 예상 피크 플레이어 동시성을 모델링하고 인프라 용량을 미리 사전 규모 조정하기 위한 사전 계획을 수행해야 합니다. 이벤트 중 플레이어 트래픽의 변동 특성으로 인해 사전 계획 및 사전 조정 활동은 기존 자동 조정 시스템을 강화하여 이벤트 중 성공 가능성을 개선하고 긍정적인 플레이어 경험을 제공할 수 있는 충분한 리소스가 있는지 확인해야 합니

다. 성능 엔지니어링 사례를 구현하여 리소스의 기준과 시스템 용량에 대한 데이터 기반 이해를 개발하면 사전 조정 활동 및 자동 조정 구성을 안내하는 데 도움이 됩니다. 운영 런북을 개발하여 프로세스의 일관성을 제공합니다.이 사전 계획 및 플레이어 수요에 대한 응답성은 라이브 서비스 게임에 특히 중요합니다. 라이브 서비스 게임은 장기간에 걸쳐 적극적이고 참여도가 높은 플레이어 기반을 지원하기 위해 안정적인 성능과 인프라를 유지해야 합니다.

- 플레이어 지원 요청을 수신, 조사 및 응답하기 위한 운영 모델 설정: 시작 후 게임 관련 불만 사항 및 문제에 대한 보고서를 모니터링합니다. 안전하고 효과적인 방식으로 플레이어와 상호 작용할 수 있는 적절한 시스템을 구현하여 커뮤니티 포럼, 소셜 미디어, 이메일, 티켓팅 시스템, 콜센터 또는 자동화된 채팅 봇 솔루션과 같은 플레이어 문제를 적절하게 해결합니다. 이는 플레이어 기반과의 지속적인 통신, 변화하는 요구 사항을 해결하기 위한 플레이어 피드백에 대한 응답성, 수명이 긴 동안 참여 커뮤니티 유지 관리가 필요한 라이브 서비스 게임에 특히 중요합니다.

라이브 작업

GAMEOPS01: 게임의 라이브 운영(라이브 운영) 전략을 어떻게 정의하나요?

비즈니스 이해관계자와 상의하여 정의된 목표 및 성과 지표를 기반으로 게임에 대한 라이브 운영(라이브 운영) 전략을 개발합니다.

모범 사례

- GAMEOPS01-BP01 게임 목표 및 비즈니스 성과 지표를 사용하여 라이브 운영 전략 개발

GAMEOPS01-BP01 게임 목표 및 비즈니스 성과 지표를 사용하여 라이브 운영 전략 개발

게임 생산자 및 게시 파트너와 같은 비즈니스 이해관계자에게 문의하여 게임의 목표 및 성능 지표를 결정합니다. 이를 통해 유지 관리 기간, 소프트웨어 및 인프라 업데이트 일정, 시스템 신뢰성 및 복구 가능성 목표를 정의하는 등 게임을 관리하는 방법에 대한 계획을 개발할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

또한 이러한 지표는 게임 수명 주기의 어느 단계에서 라이브 운영 스트림(Live Ops)을 통합하여 게임 상태를 모니터링하고, 직접적인 게임 피드백을 수집하고, 간소화되고 자동화된 릴리스 프로세스를 구

축해야 하는지 결정하는 데 도움이 될 수 있습니다. 예를 들어 새 게임은 활성 플레이어 수, 수익 또는 다른 지표 집합으로 측정되는 특정 규모에 도달할 때까지 기다렸다가 전용 라이브 운영 팀을 설정할 수 있습니다. 기존 게임 개발 스튜디오에는 이미 다른 게임에 대한 라이브 운영 경험이 있을 수 있으므로 새 게임을 온보딩하기만 하면 됩니다.

구현 단계

- 게임 인프라가 효과적으로 지원할 수 있어야 하는 플레이어 동시성(CCU) 및 일별 및 월별 활성 사용자(DAU 및 MAU) 대상, 인프라 예산, 재무 대상 및 플레이어 참여를 높이기 위한 콘텐츠 및 기능 릴리스 빈도와 같은 기타 성능 목표를 정의할 수 있습니다. 이러한 목표와 지표는 효율적인 운영에 필요한 게임 설계, 릴리스 관리, 관찰성 및 지원에 대한 결정에 도움이 됩니다.
- 게임은 릴리스 중에 가동 중지 없이 매월 한 번 이상 새 콘텐츠 업데이트를 릴리스하는 것을 목표로 할 수 있습니다. 이 정보는 릴리스 배포 전략을 정의하고 한 달 내내 다른 시간에 가동 중지가 필요하고 가용성 SLA에 기여할 수 있는 필수 유지 관리 일정을 조정하는 데 도움이 됩니다.

계정 구조

GAMEOPS02: 게임 환경을 호스팅하기 AWS 계정 위해를 어떻게 구성하나요?

다중 계정 전략을 구현하여 다양한 게임 환경을 격리하고 보안, 운영 효율성 및 확장성을 개선합니다. AWS Organizations 를 사용하여 계정 계층 구조를 관리하고, 계정에 가드레일을 적용하고, 배포된 리소스에 태그 정책 및 태그 지정을 적용합니다.

모범 사례

- GAMEOPS02-BP01 여러 게임과 애플리케이션을 자체 계정으로 격리하기 위한 다중 계정 전략 채택
- GAMEOPS02-BP02 리소스 태그 지정을 사용하여 인프라 리소스 구성

GAMEOPS02-BP01 여러 게임과 애플리케이션을 자체 계정으로 격리하기 위한 다중 계정 전략 채택

각 환경의 보안, 격리 및 운영 요구 사항을 준수하도록 인프라 배포를 안내하는 계정 구조를 설계합니다. 환경에 대한 액세스를 제한하고 필요한 AWS 서비스만 사용할 수 있도록 허용하여 환경을 격리하는 것이 필수이며, 프로덕션 환경은 잠기고 개발 및 테스트 환경은 실험을 허용할 수 있습니다. 각 환경

에서 주요 하위 시스템을 추가로 격리하고 여러 환경에서 자체적으로 AWS 계정 호스팅 및 관리하는 데 사용하는 공통 서비스를 사용하는 것이 좋습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

다양한 환경(예: 개발, 테스트, 스테이징, 프로덕션 및 공유 서비스)을 개별 환경으로 격리 AWS 하여에서 다중 계정 전략을 채택 AWS 계정하면 인시던트 범위가 줄어듭니다. 의 계층 구조를 중앙에서 관리 AWS 계정 하여 운영을 더욱 간소화하고 계정 수준 및 조직 단위 수준(OU 수준) 정책을 선택적으로 정의 및 적용하는 AWS Organizations 것이 좋습니다. 개발 및 프로덕션 워크플로 요구 사항에 맞는 적절한 OU와 AWS 계정 구조를 설계하면 비용을 최적화하고 확장성을 높일 수 있습니다.

- 다중 계정 전략 채택: 환경을 격리하여 인시던트 반경을 줄이고 운영을 간소화합니다.
- 사용 AWS Organizations: 계정을 계층적으로 관리하고, 정책을 적용하고, 중앙 집중식 거버넌스를 활성화합니다.
- 확장성 계획: 세분화된 계정 구조를 설계하고 향후 성장을 위한 비용 절감 조치를 구현합니다.

구현 단계

에 배포된 게임 시스템은 적절한 격리를 제공하도록 논리적으로 구성된 여러 계정을 사용해야 AWS 합니다. 이렇게 하면 게임 인프라가 확장될 때 문제의 폭발 반경을 줄이고 작업을 간소화할 수 있습니다. AWS 계정 호스트 게임 인프라는 일반적으로 다음과 같은 논리적 환경으로 그룹화됩니다.

- 게임 개발 환경은 개발자가 게임용 소프트웨어 및 시스템을 개발하는 데 사용됩니다.
- 테스트 또는 품질 보증(QA) 환경은 통합 테스트, 수동 QA 및 수행해야 하는 기타 자동 테스트를 수행하는 데 사용됩니다.
- 스테이징 또는 사전 프로덕션 환경은 프로덕션으로 시작하기 전에 로드 및 연기 테스트를 수행할 수 있도록 완료된 소프트웨어를 호스팅하는 데 사용됩니다.
- 라이브 또는 프로덕션 환경은 라이브 소프트웨어 및 인프라를 호스팅하고 플레이어의 프로덕션 트래픽을 제공하는 데 사용됩니다.
- 공유 서비스 또는 도구 환경은 여러 팀이 사용하는 공통 시스템, 소프트웨어 및 도구에 대한 액세스를 제공합니다. 예를 들어 중앙 자체 호스팅 소스 제어 리포지토리와 게임 빌드 팜은 공유 서비스 계정에서 호스팅될 수 있습니다.
- 보안 환경은 클라우드 보안에 중점을 둔 팀이 사용하는 중앙 집중식 로그 및 보안 기술을 통합하는 데 사용됩니다.

의 게임 인프라의 경우 각 게임 환경(개발 AWS, 테스트, 스테이징 및 프로덕션)과 보안, 로깅 및 중앙 공유 서비스를 위한 계정에 대해 별도의 계정을 생성하는 것이 좋습니다.

일반적으로 제한된 수의 인프라 리소스를 관리하는 소규모 게임 개발 스튜디오, 일반적으로 수백 개 이하의 서버가 이러한 환경(예: 프로덕션 계정 1개, 개발 계정 1개, 스테이징 계정 1개) 각각 AWS 계정 에 대해 하나씩 생성될 수 있습니다. 그러나 시간이 지남에 따라 게임 인프라 또는 팀 크기가 증가함에 따라이 간소화된 모델은 규모가 잘 조정되지 않을 수 있습니다.

이러한 환경을 설정할 때 많은 AWS 서비스가 특정 리전 내의 전체 계정에 대해 리소스 및 API 수준 Service Quotas를 공유한다는 점을 고려하세요. 이는 계정을 논리적으로 구성하는 방법을 결정할 때 고려해야 합니다. 계정에 배포된 서비스 사용에 대한 비용 AWS 계정 만 발생합니다. 따라서 특히 게임이 성장하고 더 많은 개발자가 리소스를 구축하고 관리하기 위해 액세스해야 할 때 리소스 경합과 서비스 할당량을 효과적으로 줄일 수 있는 방법을 제공합니다.

일반적으로 수백 명의 개발자가 리소스에 액세스하는 수천 개의 서버를 운영하는 대규모 게임 개발 스튜디오에서 작업한 경험을 바탕으로 게임을 지원하는 개별 애플리케이션이 자체 개발, 테스트, 스테이징 및 프로덕션 계정을 보유하는 보다 세분화된 계정 구조를 설계하는 것이 좋습니다. 라이브 시스템을 계획하고 마이그레이션하는 복잡성으로 인해 게임을 시작한 후 AWS 다중 계정 전략을 재설계하는 것은 어렵고 시간이 많이 걸리므로 올바른 다중 계정 구조를 결정할 때 향후 규모 조정 요구 사항을 고려하세요.

AWS Organizations를 사용하여 계층 구조 및 그룹화를 설정하고 AWS 계정조직 단위(OUs)를 정의하여 서비스 제어 정책(SCPs)을 통해 공통 OU 수준 정책을 적용할 수 있습니다.는 리소스를 확장하고 확장함에 따라 환경을 AWS Organizations 중앙에서 관리하고 관리합니다. 프로그래밍 방식으로 새 계정을 생성하고 리소스를 할당하며, 계정을 그룹화하여 워크플로를 구성하고, 거버넌스를 위해 계정 또는 그룹에 정책을 적용하고, 계정에 대한 단일 결제 방법을 사용하여 결제를 간소화할 수 있습니다. 또한 Organizations는 다른 서비스와 통합되어 조직의 계정 간에 중앙 구성, 보안 메커니즘, 감사 요구 사항 및 리소스 공유를 정의할 수 있습니다.

AWS Control Tower는 랜딩 존이라고 하는 안전한 다중 계정 환경을 설정하고 관리하는 간단한 방법을 제공합니다. Control Tower는를 사용하여 AWS Organizations랜딩 존을 생성하여 수천 명의 고객이 클라우드로 이동할 때 작업한 AWS경험을 기반으로 지속적인 계정 관리 및 거버넌스와 구현 모범 사례를 제공합니다. AWS Config , AWS Trusted Advisor, AWS Security Hub CSPM는 계정의 위생을 집계하거나 중앙 집중식으로 볼 수 있는 서비스입니다.

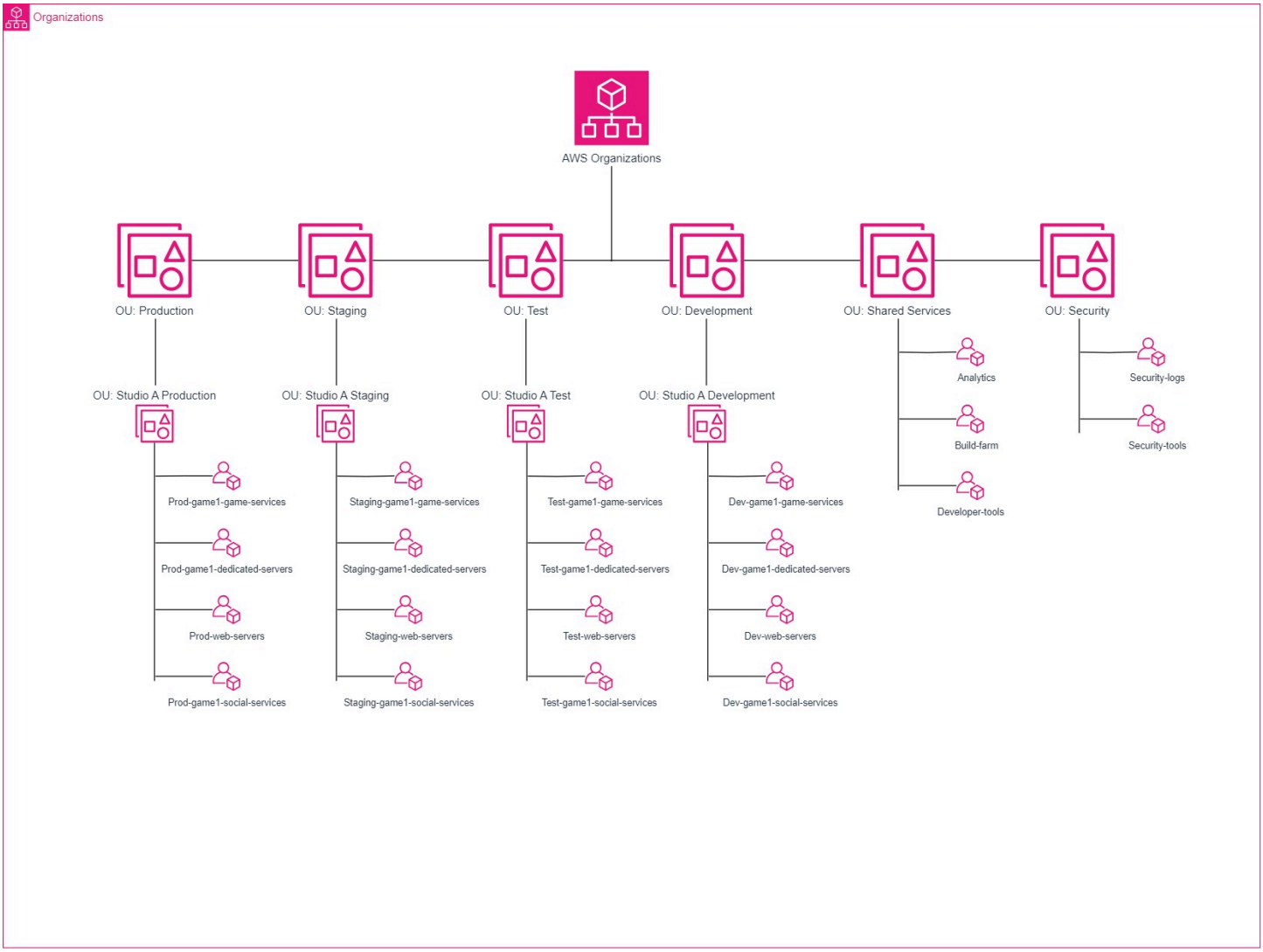
이 격리는 각 게임 환경에 대한 사용자 지정 또는 개별 권한과 가드레일을 설정하는 데 도움이 됩니다. 프로덕션 계정에는 필요한 가드레일, 액세스 제한, 모니터링 및 알림, 보안 도구가 있어야 하지만 비프로덕션 계정에는 동일한 수준의 가드레일 및 권한이 필요하지 않을 수 있습니다. 비프로덕션 환경을 자

동화하여 몇 시간 후에 리소스를 종료하고 비용을 절감할 수 있습니다. 이러한 세분화 수준에서 계정을 분리하면 게임을 지원하는 각 환경의 인프라 비용을 쉽게 모니터링할 수 있습니다.

다음은 AWS Organizations 및 조직 단위(OUs)를 사용하여 논리적으로 별도의 환경과 스튜디오 AWS 계정 로 그룹화하는 게임 회사의 다중 계정 구조의 예입니다. 이 예제에서는 OUs 사용하여 환경을 기반으로 계정을 그룹화한 다음 환경을 운영하는 스튜디오를 기반으로 계정을 그룹화합니다. 이는 중첩 계층 구조를 생성하여 별도의 애플리케이션과 게임을 환경 내 자체 계정에 배포할 수 있도록 하는 방법(OUs)을 보여줍니다. 이는 여러 게임을 개발하고 운영할 때 유용할 수 있습니다. 다중 계정 전략을 구성하는 데 고려할 수 있는 추가 전략에 대해 알아보려면이 원칙의 리소스 섹션에 제공된 설명서 및 백서를 참조하세요.

위의 논의를 기반으로 아래 샘플 다이어그램은 4단계(개발, 테스트, 스테이징 및 프로덕션)로 구성된 개발 파이프라인이 있는 게임 스튜디오(조직)를 가정합니다. 특정 게임(game1)의 경우 각 환경(OU)에는 게임 서비스, 전용 게임 서버, 소셜 서비스 및 웹 서버에 AWS 계정 대한 개별가 있습니다. 각에서 실행되는 리소스는 해당 하위 시스템과 AWS 계정 관련이 있습니다. 일반적으로 이러한 종류의 개발 파이프라인을 사용하는 모든 개별 게임은이 파이프라인 또는 유사한 구조를 복제합니다 AWS 계정.

이러한 게임 중심 환경 OUs 외에도 공유 서비스 OU 및 보안 OU도 있습니다. 이러한 OUs는 각 개별 게임이 아닌 조직 전체여야 합니다. 이렇게 하면 게임은이 예제와 같이 개발 도구, 데이터 및 분석에 공유 서비스를 사용합니다. 그런 다음 애플리케이션 및 시스템 로그를 보안 OU의 로그 AWS 계정 설정으로 전송합니다.



게임 환경의 계정 구조 예제

GAMEOPS02-BP02 리소스 태그 지정을 사용하여 인프라 리소스 구성

에서 인프라 리소스를 효과적으로 관리하고 추적하려면 적절한 리소스 태그 지정 및 그룹화를 AWS사용하여 각 리소스의 소유자, 프로젝트, 애플리케이션, 비용 센터 및 기타 데이터를 식별합니다. 태그가 지정된 리소스는 운영 지원을 지원하는 리소스 그룹을 사용하여 함께 그룹화할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

태그 지정 정책을 정의합니다. 일반적인 전략에는 팀 이름 또는 개인 이름, 게임, 애플리케이션 또는 프로젝트 이름, 스튜디오 이름, 환경(개발 또는 프로덕션 등), 리소스 역할(예: 데이터베이스 서버, 웹 서버, 전용 게임 서버, 앱 서버 또는 캐시 서버)과 같은 리소스 소유자를 식별하기 위한 리소스 태그가 포

함됩니다. 비즈니스 및 IT 요구 사항을 지원하기 위해 다른 태그를 추가할 수 있습니다. AWS Config는 리소스 생성 및 업데이트 시간에 태그 지정 정책을 적용할 수도 있습니다. 태그 및 리소스 그룹은 AWS Management Console AWS CLI, 및 API 작업에서 사용할 수 있습니다.

구현 단계

- 리소스에 태그를 지정하여 소유자, 프로젝트, 앱, 비용 센터 및 기타 관련 데이터를 식별합니다.
- 소유자, 프로젝트, 스튜디오, 환경 및 리소스 역할에 대한 태그를 포함한 태그 지정 정책을 구현합니다.
- AWS Config 를 사용하여 태그 지정 정책을 적용하고 AWS Management Console, , CLI 및 API를 통해 태그를 관리합니다.

게임 배포

GAMEOPS03: 게임 배포를 어떻게 관리하나요?

재사용된 구성 요소를 철저히 검증하고, 정기적인 성능 엔지니어링을 수행하고, 개발 수명 주기 전반에 걸쳐 정기적인 로드 테스트를 구현하여 게임 배포를 관리합니다.

모범 사례

- GAMEOPS03-BP01 게임에서 재사용하기 전에 기존 코어 게임 시스템 및 인프라를 검증하고 테스트합니다.
- GAMEOPS03-BP02 모든 릴리스 전에(또는 최소한 메이저 릴리스의 경우) 성능 엔지니어링 수행
- GAMEOPS03-BP03 로드 테스트를 조기에 자주 수행
- GAMEOPS03-BP04 플레이어에게 미치는 영향을 최소화하는 배포 전략 채택
- GAMEOPS03-BP05 피크 요구 사항을 지원하는 데 필요한 사전 규모 인프라

GAMEOPS03-BP01 게임에서 재사용하기 전에 기존 코어 게임 시스템 및 인프라를 검증하고 테스트합니다.

조직은 이전 게임의 기존 구성 요소와 소스 코드를 재사용하여 개발 시간과 비용을 절감하는 경향이 있습니다. 이러한 레거시 구성 요소 및 코드는 철저한 검토를 거치지 않거나 자세한 통합 테스트를 거치지 않고 대신 과거 성능에 의존할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

재사용은 생산성 향상에 도움이 되지만 과거 성능 및 안정성 문제를 새 프로젝트에 다시 도입할 위험도 초래할 수 있습니다. 따라서 이전 게임의 기존 구성 요소와 소스 코드를 재사용할 때는 강력한 테스트를 구현해야 합니다.

구현 단계

- 재사용된 코드 및 구성 요소 식별: 이전 게임에서 재사용되는 소스 코드, 라이브러리 및 구성 요소를 카탈로그화합니다. 적극적으로 유지 관리되는 코드와 더 이상 사용되지 않는 코드를 명확하게 구분합니다.
- 원래 동작 및 알려진 문제 문서화: 원래 성능 특성, 기능 제한, 재사용된 구성 요소와 관련된 알려진 버그 또는 프로덕션 인시던트를 기록합니다.
- 철저한 코드 검토 수행: 재사용된 구성 요소, 특히 과거에 문제가 있었거나 제대로 문서화되지 않은 구성 요소에 대한 자세한 기술 검토를 수행합니다.
- 고위험 레거시 구성 요소 교체 또는 리팩터링: 프로덕션 환경에서 해결 방법에 의존하는 대신 문제 이력이 있거나 더 이상 유지 관리할 수 없는 레거시 구성 요소를 교체하거나 업데이트하는 것을 우선시합니다.
- 통합 및 호환성 테스트 수행: 새 게임 시스템의 컨텍스트 내에서 재사용된 구성 요소를 검증합니다. 새 모듈, 도구 및 APIs와 제대로 상호 작용하는지 확인합니다.

GAMEOPS03-BP02 모든 릴리스 전에(또는 최소한 메이저 릴리스의 경우) 성능 엔지니어링 수행

성능 엔지니어링은 애플리케이션의 성능을 더욱 개선할 수 있는 최적화 기회를 찾기 위해 앱의 여러 주요 운영 지표를 모니터링하는 프로세스입니다. 이는 테스트로 시작하여 코드, 종속성, 관련 프로세스, 호스트 운영 체제 및 기본 인프라를 최적화하는 반복 프로세스입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

앱 성능에 대한 심층 분석을 수행하려면 앱 코드에 애플리케이션 성능 모니터링(APM) 또는 디버깅 도구를 통합하여 문제를 격리하고 앱 흐름 전반에서 이상에 대한 동작을 추적하여 문제 해결 시간을 단축할 수 있습니다. 또한 APM 도구는 성능이 느린 메서드와 외부 작업을 식별할 수 있습니다.

AWS X-Ray는 성능 병목 현상 식별, 프로덕션 오류 분석 및 디버깅과 같은 성능 엔지니어링 활동을 통해 개발자를 지원합니다. X-Ray를 사용하여 애플리케이션과 기본 서비스가 어떻게 작동하는지 이해하고 성능 문제 및 오류의 근본 원인을 식별하고 해결할 수 있습니다. 애플리케이션과 인프라가 합성 플레이어 트래픽과 함께 점진적으로 로드되는 여러 번의 로드 테스트를 통해 다양한 시스템 병목 현상, 앱 오류, 예외, OS 문제 및 기타 QA 테스트 중에 발견되지 않았을 수 있는 기타 문제가 식별됩니다.

게임 출시, 콘텐츠 릴리스, 프로모션 및 주요 게임 내 이벤트와 같은 중요한 이벤트의 경우 Countdown을 사용하세요. AWS Countdown은 게임 전문가가 구축한 플레이북을 기반으로 구현 지침을 제공하여 운영 준비 상태를 확인하고 잠재적 위험을 완화하며 용량 요구 사항을 계획합니다.AWS 또한 Countdown에는 인프라를 최적화하기 위한 엔지니어와 같은 향상된 지원 및 옵션을 제공하는 프리미엄 지원 옵션이 있습니다.

구현 단계

- 성능 엔지니어링에는 애플리케이션의 코드, 프로세스, 운영 체제 및 인프라가 예상대로 작동하는지 확인하기 위한 주요 운영 지표 평가 및 모니터링이 포함됩니다. 또한 사전 프로덕션 검토는 시뮬레이션된 사용량의 다양한 수준에서 기준 성능을 정의하는 데 도움이 됩니다.
- sar, top, vmstat, sysstat, netstat, Performance Monitor와 같은 시스템 도구를 사용하여 사용률, 서비스, I/O, 프로세스 등과 같은 주요 지표를 검색하고 추적합니다.
- 와 같은 APM 도구를 사용하여 애플리케이션의 성능 및 동작을 추적 AWS X-Ray 하여 문제를 격리하고, 병목 현상을 식별하고, 프로덕션 오류를 디버깅합니다.
- 게임 출시와 같은 중요한 이벤트의 경우 아키텍처 및 운영 지침, 온디맨드 운영 지원, 위험을 식별하고 완화 조치를 계획하려면 카운트다운(IEM)을 AWS 구독하세요.

GAMEOPS03-BP03 로드 테스트를 조기에 자주 수행

로드 테스트는 시스템의 실제 트래픽을 시뮬레이션하여 신뢰성과 성능을 평가하는 프로세스입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

로드 테스트는 리소스의 성능 기준을 개발하고 시스템의 용량을 이해하는 데 있어 핵심 요소입니다.이 요소는 재무 예측, 아키텍처 설계, 리소스 할당, 자동화된 조정 구성 및 출시 후 조정 전 활동을 안내할 수 있습니다. 추가 혜택은 다음과 같습니다.

- 최적화된 인프라: 리소스가 초과되거나 과소 프로비저닝될 수 있습니다. 필요한 리소스를 이해하면 비용이 절감되고 관리할 인프라가 줄어듭니다.

- 확장성 준비: 특정 메커니즘과 기능은 사용자를 게임으로 빠르게 유도할 수 있습니다. 규모를 조정하는 시기와 방법을 아는 것은 수요 증가를 적절하게 충족하고 플레이어를 잃는 것의 차이일 수 있습니다. 로드 테스트 결과를 사용하여 다양한 조정 수준에서 시스템 임계값, 알림 포인트 및 중요 알림 포인트를 사용하여 런북을 준비합니다.
- 더 높은 품질의 코드: 서비스 간 과도한 크로스톡, 배치되지 않은 데이터베이스 호출, 비효율적인 알고리즘, 메모리 누수 및 서비스 성능 저하 문제와 같은 문제가 대규모로 식별하기 더 간단한 경우가 있습니다.
- 동작 검증: 다양한 종류의 실패를 테스트에 주입하면 시스템의 예상 동작을 검증하거나 수정해야 하는 오류 처리 문제를 발견할 수 있습니다.

개발자는 개발 프로세스 전반에 걸쳐 여러 지점에서 로드 테스트를 수행하는 것이 가장 좋습니다. 각 단계에서는 다양한 이점을 얻을 수 있기 때문입니다. 초기에는 변경을 수행하는 것이 더 저렴하고 간단하면서도 아키텍처 결정과 리팩터링 작업을 안내합니다. 각 스프린트 또는 반복이 끝나면 최신 기능을 사용하여 애플리케이션의 성능을 검증합니다.

프로덕션에 배포하기 전에 예상되는 실제 사용 패턴을 시뮬레이션하는 대규모 로드 테스트는 프로덕션 워크로드를 처리하는 시스템의 능력을 확인합니다. 배포 후 정기적인 로드 테스트는 시스템 성능을 모니터링하고 시간 경과에 따라 발생할 수 있는 변경 사항 또는 병목 현상을 식별합니다.

플레이어 트래픽을 시뮬레이션하려면 게임 클라이언트 흐름을 에뮬레이션하고 게임 백엔드와 트랜잭션하여 실제 플레이어 동작을 시뮬레이션하는 경량 클라이언트 또는 봇이 필요합니다. 이 데이터는 일반적으로 게임 플레이 로그와 인간 기반 QA 테스트에서 생성된 데이터를 통해 캡처될 뿐만 아니라 실제 플레이어가 게임의 조기 액세스 빌드를 플레이하도록 초대되는 실제 제한적 규모 알파 또는 베타 테스트를 통해 캡처됩니다.

향후 발생할 수 있는 장애 문제를 해결하고 향후 로드 테스트를 비교할 수 있는 성능 지표를 유지하려면 운영 실행서에 시스템 동작을 기록하는 것이 중요합니다. 또한 인간 QA 담당자가 로드 테스트 중에 게임을 테스트하도록 하는 것이 좋습니다. 봇이 식별하지 못하고 지표가 반영되지 않는 문제를 발견할 수 있기 때문입니다.

AWS Fault Injection Service는 애플리케이션의 성능, 관찰성 및 복원력을 쉽게 개선할 수 있는 결함 주입 실험을 실행하기 위한 완전 관리형 서비스입니다. 결함 주입 실험은 카오스 엔지니어링에 사용되며, 이는 CPU 또는 메모리 소비의 갑작스러운 증가, 시스템 응답 방식 관찰, 개선 구현과 같은 중단 이벤트를 생성하여 테스트 또는 프로덕션 환경에서 애플리케이션에 스트레스를 가하는 관행입니다. 결함 주입 실험은 팀이 분산 시스템에서 찾기 어려운 숨겨진 버그, 모니터링 사각지대 및 성능 병목 현상을 발견하는 데 필요한 실제 조건을 생성하는 데 도움이 됩니다.

구현 단계

- Kubernetes-Bases 게임 로드 테스트 지침을 사용하여 분산 로드 테스트 환경을 설정합니다.
- 제공된 배포 파일을 사용하여 EKS 클러스터 내에서 Locust 제어 및 작업자 포드를 사용자 지정하고 배포하여 확장 가능하고 관리 가능한 로드 생성을 지원합니다.
- 로드 테스트 중 시스템 동작과 지표를 운영 런북에 기록하여 향후 문제 해결을 지원하고 성능 기준을 설정합니다.
- 결함 주입 실험을 사용하여 실제 중단을 시뮬레이션하고 시스템 성능, 관찰성 및 복원력에서 숨겨진 문제를 발견합니다.

GAMEOPS03-BP04 플레이어에게 미치는 영향을 최소화하는 배포 전략 채택

플레이어를 게임에서 차단하는 가동 중지 시간을 최소화하는 게임 소프트웨어 및 인프라에 대한 배포 전략을 통합합니다. 특정 유형의 업데이트에는 게임 클라이언트에 새 업데이트를 설치해야 할 수 있지만 배포 중에 가동 중지 시간이 필요하지 않도록 게임을 설계합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임 배포 전략을 개발할 때 고려해야 할 가장 중요한 단계 중 하나는 게임 인프라를 관리하는 방법을 결정하는 것입니다. Hashicorp의 AWS CloudFormation 또는 Terraform과 같은 코드형 인프라(IaC) 도구를 사용하여 게임 인프라를 관리하여 환경 준비 중에 인적 오류를 줄입니다. 인프라 템플릿은 자동화된 파이프라인에 배포하고 테스트할 수 있으므로 다양한 게임 환경의 구성에 일관성을 유지할 수 있습니다.

게임에 사용할 수 있는 몇 가지 배포 전략이 있습니다.

롤링 대체

배포를 위한 롤링 대체의 주요 목표는 게임을 종료하지 않고 플레이어에게 영향을 주지 않고 릴리스를 수행하는 것입니다. 수행할 업그레이드 또는 변경 사항은 이전 버전과 호환되며 이전 버전의 시스템과 인접하여 작동하는 것이 중요합니다.

이 배포에서는 서버 인스턴스가 업데이트된 버전을 실행하는 인스턴스로 점진적으로 교체(대체 또는 롤아웃)됩니다. 이 롤링 대체는 몇 가지 방법으로 수행할 수 있습니다. 예를 들어 전용 게임 서버 플릿

에 롤링 업데이트를 구현하기 위해 일반적인 접근 방식에는 배포된 새 게임 서버 빌드 버전이 포함된 새 Auto Scaling EC2 인스턴스 그룹을 생성한 다음이 새 서버 플릿에서 호스팅되는 게임 세션으로 플레이어를 점진적으로 라우팅하는 것이 포함됩니다. 새 게임 서버 빌드를 사용하기 위한 사전 조건으로 필요한 연결된 게임 클라이언트 업데이트가 있는 경우 검증 검사를 포함하여이 새 게임 클라이언트 업데이트가 설치된 플레이어만 이러한 게임 세션으로 라우팅되는지 확인해야 합니다.

이전 게임 서버 빌드 버전이 포함된 서버 플릿(예: EC2 Auto Scaling 그룹)은 일반적으로 게임 운영 팀이이 프로세스를 자동화할 수 있도록 개별화된 서버 지표를 설정하여 활성 플레이어 세션이 고갈된 후에만 서비스에서 제거됩니다. 또는 인프라 양과 롤링 배포 수행 시간을 줄이기 위해 기존 프로덕션 인스턴스를 서비스에서 제거하고 새 게임 서버 빌드로 업데이트한 다음 프로덕션 플릿에 다시 배치하는 대체 접근 방식을 수행할 수 있습니다. 이 접근 방식은 필요한 인프라의 양을 줄이지만 서버를 교체함에 따라 플레이어에게 사용 가능한 라이브 게임 서버의 수가 줄어들기 때문에 위험도 증가합니다.

이 모델은 게임 플레이를 호스팅하지 않는 데이터베이스, 캐시 및 애플리케이션 서버와 같은 백엔드 서비스에 대한 롤링 배포를 수행하는 데에도 사용할 수 있습니다. 이러한 서비스가 여러 클러스터링된 인스턴스에서 가용성이 높은 방식으로 배포되는 한, 이러한 서비스에 대한 배포의 복잡성은 전용 게임 서버에 대한 배포보다 작아야 합니다.

블루/그린 배포

게임에서 블루/그린 배포의 주요 목표는 가동 중지 시간을 최소화하는 동시에 문제가 식별되면 이전 배포로 안전하게 롤백할 수 있도록 하는 것입니다. 게임 백엔드의 두 버전이 호환되고 플레이어에게 동시에 서비스를 제공할 수 있는 배포에 적합합니다.

블루/그린 배포 전략에서는 두 개의 동일한 환경(블루 및 그린)이 설정됩니다. 기존 게임 버전에는 파란색 레이블이 지정되고 배포 대상인 새 게임 버전에는 녹색 레이블이 지정됩니다. 그린 환경을 마이그레이션할 준비가 되면 장애 복구가 필요한 경우 이전 환경(블루)을 사용할 수 있도록 유지하면서 트래픽을 그린 환경으로 넘기도록 라우팅 계층을 구성할 수 있습니다. 이 시나리오에서 라우팅 업데이트를 수행하려면 매치메이킹 서비스를 업데이트하여 게임 세션을 새 플릿으로 전송하도록 구성하거나 게임 백엔드 서비스의 경우 서비스에 대해 Amazon Route 53의 DNS 레코드를 업데이트하거나 새 대상 그룹에 트래픽을 전송하도록 애플리케이션 로드 밸런서 가중치를 이동해야 할 수 있습니다.

블루/그린 배포 전략의 단점 중 하나는 배포를 수행하는 데 필요한 추가 인프라로 인해 대기 환경의 고유한 비용입니다. 이 추가 인프라 비용을 완화하는 옵션은 이미 프로덕션에 배포된 동일한 서버에 새 게임 소프트웨어가 배포되는 블루/그린 배포 변형을 채택하는 것을 고려하는 것입니다. 이 시나리오에서는 기존 블루 서버 프로세스와 함께 새 소프트웨어로 새 그린 서버 프로세스를 시작할 수 있으며, 별도의 물리적 인프라가 아닌 서버 프로세스 간에 전환이 이루어집니다. 또한이 접근 방식은 클라우드에서 새 서버가 시작될 때까지 기다릴 필요 없이 대규모 인프라에서 게임 배포 속도를 높일 수 있습니다. 이 배포 접근 방식에 대한 모범 사례는 블루/그린 배포를 참조하십시오 AWS.

카나리아 배포

Canary 배포는 게임의 초기 알파 또는 베타 빌드를 릴리스하거나, 새로운 게임 모드, 맵 또는 챌린지와 같은 게임 기능을 제한되거나 작은 플레이어 프로덕션 세트에 릴리스하는 데 전략을 적용할 수 있으므로 게임 개발자에게 유용합니다. 이러한 배포를 카나리아라고 합니다. 릴리스에는 추가 추적 및 보고 기능이 있을 수 있으므로 실제 플레이어가 해당 게임 또는 기능을 플레이하면 게임 플레이 원격 측정이 수집되어 이상 및 문제가 있는지 분석됩니다.

새로운 기능의 경우 플레이어에게 이에 대해 지속적으로 알리지 않으며, 게임 원격 측정은 플레이어에게 문제가 발생하고 릴리스를 롤백해야 하는지 여부를 결정하는 데 사용되는 기본 소스입니다. 동시에 중요한 문제가 식별되지 않으면 추가 데이터를 위해 더 많은 플레이어에게 기능을 추가로 배포할 수 있습니다. 플레이어에게 알림을 보내면 자신의 경험에 대한 정기적인 피드백을 제공하도록 요청할 수 있습니다. 이러한 테스트 활동은 라이브 운영 팀에서 조정하는 것이 이상적입니다.

전략으로 canary 배포를 표준 릴리스에 사용하여 플레이어가 새로운 기능을 점진적으로 사용할 수 있도록 할 수도 있습니다. 표준 블루/그린 환경에 비해 잠재적인 이점은 전체 규모의 두 번째 환경이 필요하지 않다는 것입니다. 새로운 축소 환경의 용량은 새로운 기능에 온보딩할 플레이어 수를 결정합니다. 플레이어를 더 추가하기 전에 용량을 적절하게 조정해야 합니다. 이 사용자 지정 블루/그린 기술은 표준 블루/그린보다 비용이 비교적 적게 들 것으로 예상되지만 카나리 배포의 롤링 대체 기술보다 비용이 더 많이 들 수 있습니다.

프로덕션 환경에서 단일 canary만 실행하고 데이터와 피드백에 집중합니다. 여러 canary가 배포되면 문제 해결 및 프로덕션 문제 격리가 복잡해지고 수집 중인 데이터 세트 및 피드백의 품질이 저하됩니다.

카나리아의 변형은 하나 이상의 실험(일반적으로 UI 테스트)이 대상 배포를 통해 실행되는 경우입니다. 여기서 게임 백엔드 서버 세트 하나는 한 버전의 기능을 제공하고 동일한 크기의 다른 세트는 동일한 기능의 다른 버전을 제공합니다. 이를 위한 추가 또는 특수 인프라는 생성되지 않으며, 선택한 백엔드 서버 주머니만 이러한 업데이트를 수신합니다. 실험의 결과는 플레이어가 동일한 기능의 각 버전에 어떻게 반응하는지 관찰하고, 전반적으로 좋아요 또는 싫어요에 대한 합의가 있는지 확인하고, 사용성 또는 기능과 관련하여 식별된 문제가 있는지 관찰하는 것입니다. 이러한 전략적 실험을 A/B 테스트라고도 하며, 전체 프로세스를 A/B 테스트라고 합니다. 이러한 실험이 완료되면 테스트에 사용되는 서버에서 게임 백엔드 시스템의 현재 버전으로 되돌리기 전에 필요한 테스트 데이터가 수집됩니다.

레거시 기존 배포

기존 배포 방식에서는 예약된 유지 관리 기간 동안 게임 백엔드 내의 서버 인스턴스가 최신 코드 빌드로 업데이트되기 전에 게임이 종료되고 연결된 플레이어가 삭제되거나 드레이닝됩니다. 이 배포는 수행할 때마다 플레이어에게 영향을 미치므로 플레이어에게 일정에 앞서 알림을 보내야 합니다. 따라서 이 모델은 플레이어에게 가장 큰 영향을 미치므로 가능하면 피해야 합니다.

게임 업데이트가 배포된 후 게임이 다시 열릴 때까지 기다리는 플레이어에게 게임을 열기 전에 게임을 연기 테스트할 수 있습니다. 이로 인해 플레이어가 짧은 시간 내에 로그인하고 플레이하려고 할 때 트래픽이 급증할 수 있습니다. 따라서 게임이 이러한 트래픽 급증을 처리하도록 설계되지 않은 경우 플레이어가 게임에 배치로 다시 참여하도록 점진적으로 허용할 수 있습니다.

또는 인프라 오버프로비저닝을 선택하여 트래픽의 급증을 지속하고 게임 트래픽이 안정화된 후 리소스를 축소할 수 있습니다. 필요한 경우 플레이어 수가 가장 적은 사용량이 적은 시간에 이러한 유형의 배포를 수행합니다. 유지 관리가 자주 예약되고 유지 관리가 연장되면 기본적으로 플레이어 이탈 및 잠재적 수익 손실 위험이 있습니다. 또한 플레이어는 새 릴리스 이후 변경 사항을 예상하며 가동 중지 시간이 지난 후 게임에 대한 신뢰를 잃을 수 있습니다.

구현 단계

- 가동 중지 시간 최소화: 가동 중지 시간을 줄이고 플레이어를 게임에 유지하는 배포 전략을 구현합니다.
- 코드형 인프라(IaC): AWS CloudFormation 또는 Terraform과 같은 도구를 사용하여 게임 인프라를 관리하고 인적 오류를 줄입니다.
- 배포 전략: 롤링 대체, 블루/그린 및 카나리 배포 중 하나 또는 조합을 사용하여 원활한 업데이트를 제공하고 플레이어의 영향을 줄입니다.

GAMEOPS03-BP05 피크 요구 사항을 지원하는 데 필요한 사전 규모 인프라

대규모 게임 이벤트보다 먼저 인프라를 확장하여 플레이어 수요의 갑작스러운 증가를 처리할 수 있도록 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

새로운 게임 출시 외에도 라이브 게임은 일반적으로 플레이어 참여를 유지하고 개선하는 방법의 예로 게임 내 이벤트, 프로모션, 새로운 콘텐츠 및 시즌 릴리스를 실행합니다. 이러한 활동은 이벤트 또는 홍보 기간 동안 많은 양의 플레이어 트래픽을 경험합니다. 비즈니스는 이벤트에 대해 의도한 목표를 달성하거나 초과할 것으로 예상하며, 게임 인프라는 이를 통해 이를 유지하고 지원해야 합니다.

대규모 이벤트 중에 발생할 것으로 예상되는 플레이어 로드를 지원할 수 있도록 인프라를 미리 준비합니다. 준비를 위해 게임 운영 팀은 영업 및 마케팅의 이해관계자와 협력하여 과거 플레이어 동시성, 참여 지표 및 판매 데이터를 검토하여 향후 이벤트에서 발생할 예상 수요를 추정해야 합니다. 이벤트가 새로운 게임 출시를 위한 경우 게임 운영 팀은 이러한 이해관계자와 협력하여 예상 규모에 대한 현실적인 예측을 식별해야 합니다. 게임이 얼마나 성공할지 예측하기는 어려울 수 있지만, 인프라를 확장하고

테스트하여 이러한 목표를 지원할 수 있도록 모든 사람이 성공에 대한 기대치를 이해하는 것이 중요합니다.

많은 게임은 소프트 런치부터 시작하여 소수의 플레이어에게 게임을 열고 전체 퍼블릭 런칭 전에 모든 스테이지에서 플레이어를 유기적으로 조정하여 스테이지에서 런칭하기로 선택합니다. 소프트 런치 기간 동안 퍼블릭 런치에 대한 프로젝션을 구체화하면서 문제를 모니터링, 식별, 추적 및 해결합니다.

인프라 요구 사항을 올바르게 추정하려면 게임 시작 전에 프로덕션 또는 프로덕션과 유사한 스테이징 환경에서 실행되는 게임 백엔드에 대해 실행되는 로드 및 성능 테스트를 통해 데이터를 수집합니다. 이러한 테스트를 여러 번 실행하여 게임의 다양한 조건을 시뮬레이션하고 백엔드가 대부분의 조건에서 로드를 견딜 수 있는지 확인해야 합니다.

이를 위해 개발자는 게임의 다양한 워크플로를 통과하고 다양한 조건을 에뮬레이션하는 게임 플레이 봇을 작성할 수 있습니다. 이러한 테스트는 게임 백엔드의 다양한 시스템 계층을 검사하여 각 계층과 구성 요소를 테스트하고 세부 정보를 기록해야 합니다. 이러한 테스트에서 수집된 데이터를 사용하여 게임 시작 계획을 프로비저닝합니다.

애플리케이션의 가용성과 내결함성을 높여 가능하면 단일 장애 지점(SPOF)을 식별하고 제거해야 합니다. 로드 테스트를 사용하여 다양한 업스트림 및 다운스트림 계층에서 장애를 에뮬레이션하고 게임 및 기타 구성 요소 동작을 확인하여 SPOFs를 식별합니다.

게임 시작, 게임 내 이벤트 또는 프로모션 준비에 필요한 예상 인프라와 함께 온디맨드 방식으로 자동으로 규모를 조정하도록 시스템을 설정합니다. 게임 백엔드가 대규모 플레이어 트래픽을 유지할 수 있도록 조정 이벤트 임계값을 정의, 구성 및 모니터링합니다. 가변 트래픽의 경우 확장할 시간이 충분하지 않을 수 있으므로 사전 프로비저닝이 가장 좋습니다. 수동 조정은 자동화된 시스템이 리소스를 확장할 수 있는 것보다 예상보다 빠르게 수요를 높이는 초기 게임 시작 중에 필요할 수 있습니다.

에서 AWS조직은 게임 백엔드에서 사용하는 서비스에 대해 더 높은 Service Quotas를 요청해야 합니다. Service Quotas는 의도한 것보다 많은 인프라를 실수로 설정하거나 확장하지 못하도록 고객을 보호하기 위해 계정에 대해 설정됩니다. 계정에서 실행 중인 게임이 해당 리전에서 구성된 서비스 할당량의 상한에 도달하면 서비스는 프로비저닝된 할당량 및 버스트 프로비저닝 이상으로 요청을 제한합니다. 제한으로 인해 의도하지 않거나 예상치 못한 오류가 발생하여 플레이어 경험이 손상될 수 있습니다. 게임 프로덕션에서 사용하는 서비스에 대한 서비스 할당량 임계값을 모니터링, 추적 및 정기적으로 검토하여 제한을 방지합니다. 사용량이 허용 가능한 서비스 할당량 임계값을 초과하면 콘솔 지원 센터에서 지원 사례를 제기하거나 영향을 받는 계정에 로그인한 후 또는 지원 API를 사용하여 할당량 증가를 요청할 수 있습니다.

게임 출시, 콘텐츠 릴리스, 프로모션 및 주요 게임 내 이벤트와 같은 중요한 이벤트의 경우 AWS Countdown을 사용합니다. Countdown은 운영 준비 상태를 제공하고, 잠재적 위험을 완화하고, 용량 요

구 사항을 계획하기 위해 Games 전문가가 구축한 플레이북을 기반으로 구현 지침을 제공합니다.AWS Countdown에는 인프라를 최적화하기 위해 엔지니어와 같은 향상된 지원 및 옵션을 제공하는 프리미엄 지원 옵션도 있습니다.

Amazon GameLift에서 호스팅되는 게임을 시작하는 경우 시작 전 체크리스트를 검토하여 준비합니다.

구현 단계

- 미리 인프라 확장: 플레이어 수요의 갑작스러운 증가를 처리할 수 있도록 대규모 게임 이벤트에 대한 인프라를 미리 준비합니다.
- 수요 추정: 판매 및 마케팅과 협력하여 과거 플레이어 데이터와 사실적인 예측을 사용하여 예상 수요를 추정합니다.
- 로드 테스트 및 SPOF 제거: 여러 번의 로드 테스트를 수행하여 백엔드 용량을 검증하고, 단일 장애 지점을 식별하고, 자동 조정을 적절하게 구성합니다.

상태 모니터링

GAMEOPS04: 게임 상태를 어떻게 모니터링하나요?

포괄적인 계측을 구현하여 클라이언트 측 활동 로깅, 백엔드 서비스 모니터링, 오류 보고 등 플레이어에게 영향을 미치는 문제를 감지하고 추적하여 게임 상태를 모니터링합니다. Amazon CloudWatch 및 같은 AWS 도구와 AWS X-Ray타사 솔루션을 조합하여 문제를 빠르게 식별하고 해결할 수 있습니다.

모범 사례

- GAMESOPS04-BP01 플레이어에 영향을 미치는 문제를 감지하고 모니터링하도록 게임 계측

GAMESOPS04-BP01 플레이어에 영향을 미치는 문제를 감지하고 모니터링하도록 게임 계측

소셜 미디어 및 플레이어 문제 보고서에 응답하는 것 외에도 플레이어에 영향을 미치는 문제를 감지하고 조사하는 모니터링 솔루션으로 게임을 계측합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임의 모든 문제를 식별할 수 있는 테스트는 없습니다. 게임은 일반적으로 알려진 문제로 시작되며, 게임의 다음 릴리스에서 점진적으로 수정될 예정입니다. 알려진 문제와 재현 가능한 문제를 쉽게 해결하고 수정할 수 있습니다. 이러한 문제를 식별하는 데 도움이 되도록 게임 클라이언트는 다양한 전략적 위치에서 플레이어 활동 추적, 앱 로깅 및 보고를 구현하여 백엔드 팀이 클라이언트 측 문제를 식별할 수 있도록 지원해야 합니다. 이러한 문제를 조기에 찾을 수 있는 기능은 게임 개발자가 문제가 널리 퍼지기 전에 문제를 해결하고 해결하는 데 도움이 됩니다. 추적 코드에서 보고하는 데이터 및 로그에는 개인 식별 정보(PII)가 포함되어서는 안 되며, 디버깅을 지원하는 게임별 메타데이터만 포함되어야 합니다.

게임 충돌 또는 버그와 같은 문제를 감지하고 대응하기 위한 관찰성 솔루션을 구현합니다. Amazon CloudWatch Synthetics를 사용하여 플레이어 대상 백엔드 게임 서비스의 상태를 모니터링할 수 있는 canary를 생성할 수 있습니다. 를 사용하여 백엔드 서비스를 계측AWS X-Ray하여 분산 서비스 전반의 요청을 추적하고 사용자 지정 로그 및 지표를 Amazon CloudWatch로 전송할 수 있습니다.

Backtrace.io 및 Sentry와 같은 타사 솔루션은 게임에서 오류를 보고하는 데 널리 사용되는 솔루션입니다. New Relic, Splunk, Datadog, Honeycomb.io 등의 파트너의 애플리케이션 성능 모니터링(APM) 솔루션도 인기가 있습니다.

또한 게임의 라이브 운영 팀과 커뮤니티 관리자는 다양한 소셜 네트워크와 채널을 모니터링하여 공식 지원 채널 외에도 플레이어 피드백, 불만 제기 및 버그 보고서를 확인해야 합니다. 모든 게임별 수신 거부를 검토 및 재현하거나 검토를 위해 QA 팀에 전송합니다. 재현 가능한 경우 더 큰 플레이어 기반에 영향을 미치기 전에 문제 해결 및 수정을 위해 게임 개발자에게 문제를 에스컬레이션합니다.

구현 단계

- 모니터링 솔루션 구현: 모니터링 도구를 사용하여 플레이어에게 영향을 미치는 문제를 감지하고 신속하게 대응합니다.
- 플레이어 활동 및 로그 추적: 게임 클라이언트를 계측하여 플레이어 활동을 기록하고 문제를 보고하며 개인 식별 정보(PII)가 포함되어 있지 않은지 확인합니다.
- 타사 및 AWS 도구 사용: 오류 보고 및 성능 모니터링을 위해 CloudWatch, X-Ray 및 타사 솔루션과 같은 도구를 사용하고 플레이어 피드백 및 버그 보고서를 위해 소셜 미디어를 모니터링합니다.

로드 테스트.

GAMEOPS05: 게임을 로드 테스트할 때 고려해야 할 사항은 무엇입니까?

게임을 로드 테스트할 때 적절한 테스트 단계, 로드 생성 아키텍처 및 테스트 프레임워크를 고려하여 시스템 성능과 확장성을 효과적으로 평가합니다. 게임의 고유한 특성 및 개발 목표에 맞게 타이밍(초기 개발, 스프린트, 사전 프로덕션 또는 배포 후), 인프라(EC2, EKS, Fargate 또는 Lambda) 및 테스트 도구(JMeter, Locust, Grafana K6 또는 Gatling)의 올바른 조합을 선택합니다.

모범 사례

- GAMEOPS05-BP01 목표에 맞는 적절한 단계, 아키텍처 및 로드 테스트 프레임워크 선택

GAMEOPS05-BP01 목표에 맞는 적절한 단계, 아키텍처 및 로드 테스트 프레임워크 선택

게임 로드 테스트에 대한 접근 방식은 게임이 수행되는 개발 프로세스의 단계, 로드 생성 시스템 자체의 아키텍처, 로드 테스트 프레임워크 선택 등 여러 요인에 따라 크게 달라질 수 있습니다. 초기 단계이든, 반복 스프린트 중이든, 프로덕션 배포 전이든, 배포 후이든 상관없이 수행되는 시기는 테스트 작업의 목표와 초점을 형성합니다. 로드 생성 인프라의 다양한 설계에는 고유한 장단점이 있으며, 로드 테스트 프레임워크를 선택하면 테스트 프로세스에 사용할 수 있는 기능, 사용 편의성 및 통합에 큰 영향을 미칩니다. 개발 팀은 이러한 요소를 신중하게 조정하여 로드 테스트 접근 방식을 게임의 고유한 특성에 맞게 조정하고, 가장 중요한 성능 인사이트를 추출하고, 플레이어에게 원활한 경험을 제공할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

다양한 개발 단계에서 로드 테스트

개발 단계 초기에 탐색 로드 테스트를 수행하면 기본 시스템 아키텍처를 검증할 수 있습니다. 이를 통해 개발자는 광범위한 구현 작업이 완료되기 전에 게임의 인프라, 데이터베이스 설계 및 네트워크 토폴로지에 대해 정보에 입각한 결정을 내릴 수 있습니다. 로드 테스트는 위험을 식별하고 성능 기준을 생성하여 개발 수명 주기 후반부에 비용이 많이 드는 재작업 및 기술 부채의 필요성을 최소화할 수 있습니다. 또한 팀 간의 게임 성능 요구 사항에 대한 공동 이해를 촉진하여 협업과 의사 결정을 개선할 수 있습니다. 궁극적으로 초기 단계의 로드 테스트는 성능, 확장성 및 복원력이 뛰어난 게임을 위한 강력한 기반을 구축하여 전반적인 플레이어 경험을 개선하는 데 도움이 됩니다.

각 스프린트 또는 반복이 끝나면 로드 테스트는 최신 주기에 도입된 새로운 기능, 버그 수정 및 기타 변경 사항의 성능 영향을 평가할 수 있습니다. 이 목표 접근 방식을 통해 개발 팀은 최신 업데이트로 인한 회귀 또는 성능 저하를 신속하게 식별할 수 있으므로 파이프라인으로 더 전파되기 전에 이러한 문제를 해결하고 일관된 수준의 품질과 성능을 유지할 수 있습니다.

프로덕션에 배포하기 전에 강력한 로드 테스트를 통해 팀은 예상되는 실제 트래픽 및 로드 조건을 처리할 수 있는 시스템의 기능을 검증할 수 있습니다. 프로덕션 인프라 내에서 확장성 병목 현상 또는 리소스 제약 조건을 발견하고 게임 성능을 최적화하여 첫날부터 원활하고 응답성이 뛰어난 사용자 환경을 만들 수 있는 기회를 제공할 수 있습니다. 출시 전 로드 테스트에서 얻은 인사이트는 출시일 위험을 완화하고 지속적인 용량 계획에 정보를 제공할 수 있으며, 이는 게임의 장기 지속 가능성 및 확장성의 토대를 마련합니다.

이미 프로덕션 환경에 있는 게임을 로드 테스트하면 팀이 게임의 성능을 모니터링하고 시간 경과에 따라 발생할 수 있는 성능 저하 또는 성능 저하를 식별할 수 있습니다. 이를 통해 플레이어 경험에 영향을 미치고 사용자 보존에 부정적인 영향을 미치기 전에 문제를 사전에 해결할 수 있습니다. 또한 프로덕션에서의 로드 테스트는 구현된 성능 최적화 작업 또는 인프라 규모 조정의 효과를 검증합니다. 이 프로세스는 게임이 발전하고 성숙해지더라도 플레이어에게 응답성이 뛰어나고 확장 가능한 고품질 게임 환경을 제공합니다.

로드 생성 아키텍처

게임 로드 테스트를 위한 로드 생성 아키텍처의 설계는 다양한 형태를 취할 수 있으며, 각 아키텍처에는 고유한 장점과 고려 사항이 있습니다.

가장 기본적인 수준에서 자체 관리형 Amazon EC2 인스턴스는 로드 생성기 역할을 하도록 프로비저닝하고 구성할 수 있습니다. 제어 노드 및 작업자 노드 접근 방식을 사용하면 각각 자체 테스트 스크립트를 실행하고 전체가 단일 제어 인스턴스에서 관리하는 여러 로드 생성 인스턴스를 설정할 수 있습니다. 아키텍처는 추가 작업자 노드를 가동하여 복잡성을 높이지 않고 확장하고 더 많은 로드를 생성할 수 있지만,이 실습 접근 방식을 사용하려면 팀이 기본 인프라의 프로비저닝, 구성 및 관리를 처리해야 합니다.

보다 확장 가능하고 오케스트레이션된 접근 방식을 위해 Amazon EKS Kubernetes 클러스터를 사용하여 컨테이너 기반 로드 에이전트 플릿에 로드 테스트 워크로드를 관리하고 배포할 수 있습니다. Kubernetes 자동 조정 기능을 사용하여 로드 생성 포드의 규모 조정을 처리할 수 있으며 팀 자체는 포드를 호스팅하는 클러스터의 기본 EC2 인스턴스를 구성하고 관리합니다.

또는의 서버리스 특성은 필요한 확장성과 유연성을 제공하면서 인프라 관리를 추상화하여 로드 테스트 설정을 가속화하고 단순화할 AWS Fargate 수 있습니다. 온프레미스, 로드 생성 Kubernetes 클러스터가 이미 있지만 추가 용량이 필요할 수 있는 하이브리드 솔루션의 경우 EKS Anywhere는 두 클러스터를에서 하나로 관리할 수 있습니다 AWS Management Console.

요구 사항 및 목표에 따라 AWS Lambda 함수를 사용할 수도 있습니다. Lambda 함수는 추가 리소스를 프로비저닝하고 관리할 필요 없이 비교적 간단하게 설정하고 확장할 수 있습니다. 또한 다른 AWS 서비스와의 심층 통합으로 인해 더 복잡하고 동적인 테스트 시나리오를 생성할 수 있습니다. 그러나 Lambda 함수에는 동시 함수 및 런타임(15분)에 대한 제한이 있으므로 달성할 수 있는 로드 테스트의

규모와 길이가 제한될 수 있습니다. 콜드 스타트 지연 시간은 결과의 정확도에도 영향을 미칠 수 있으며, Lambda의 리소스 제한은 매우 까다로운 로드 테스트 워크로드에 적합하지 않을 수 있습니다.

사전 구축된 솔루션을 사용하려는 스튜디오는 에서 분산 로드 테스트를 AWS 사용할 수 있습니다. 이 솔루션은의 Amazon ECS AWS Fargate 를 사용하여 수만 명의 연결된 사용자로 구성된 시뮬레이션을 실행할 수 있는 컨테이너를 배포합니다. 이를 사용하여를 사용하여 IAC 방식으로 로드 테스트 인프라를 빠르게 시작할 수 있습니다 AWS CloudFormation.

로드 테스트 프레임워크

두 로드 테스트 프레임워크가 동일하게 빌드되지 않습니다. 테스트 생성을 위한 직관적인 그래픽 인터페이스가 있는 것도 있고, 명령줄을 기반으로 하는 것도 있습니다. 한 도구는 유연하고 성능이 뛰어나지만 구성 및 관리에 시간과 노력이 필요할 수 있으며, 다른 도구는 서버리스이지만 생성 및 실행할 수 있는 테스트에서 제한될 수 있습니다. 일부는 현장에서 입증되지 않은 상태에서 대규모 커뮤니티와 많은 자습서를 즐길 수 있으며, 프로덕션 환경에서 전투 테스트를 거쳤지만 커뮤니티 지원이나 문서가 부족한 다른 사람들과는 명확하게 대조됩니다. 자신과 팀에 적합한 균형을 이루는 프레임워크를 선택합니다. 몇 가지 인기 옵션은 다음과 같습니다.

- Apache JMeter: 강력한 기능 세트와 사용 편의성으로 인해 인기 있는 Java 기반 오픈 소스 로드 테스트 프레임워크입니다. 복잡한 사용자 시나리오, 다양한 지원 프로토콜, 포괄적인 보고 및 검증된 트랙 레코드를 시뮬레이션하는 기능을 갖춘 JMeter는 로드 테스트를 위한 신뢰할 수 있는 선택입니다.
- Locust: 이벤트 기반 아키텍처를 기반으로 구축된 현대적 분산 로드 테스트 프레임워크로, 성능이 뛰어나면서도 리소스 효율적입니다. 테스트는 Python으로 작성되므로 수천 개의 강력한 타사 라이브러리를 활용하는 동시에 친숙하고 읽기 쉬운 유연한 테스트 시나리오가 가능합니다.
- Grafana K6: 사용 편의성과 고급 기능을 결합한 강력한 로드 테스트 프레임워크입니다. 분산 로드 생성, 유연한 스크립팅, 데이터 시각화를 위한 Grafana와의 원활한 통합을 지원하는 Grafana K6는 매력적인 선택입니다.
- 개틀링: 성능과 확장성으로 알려진 오픈 소스 로드 테스트 프레임워크입니다. Scala 기반 도메인별 언어(DSL)를 통해 개발자는 간결하고 유지 관리 가능한 로드 테스트 스크립트를 생성할 수 있으며, 강력한 보고 및 분석 기능은 테스트 중인 시스템에 대한 자세한 인사이트를 제공합니다.

구현 단계

- 로드 테스트 단계: 다양한 개발 단계(초기 개발, 스프린트, 사전 프로덕션 및 배포 후)에서 로드 테스트를 수행하여 시스템 성능을 검증하고 문제를 식별합니다.
- 로드 생성 아키텍처: 확장성 요구 사항, 관리 기본 설정 및 특정 테스트 요구 사항에 따라 적절한 로드 생성 아키텍처(EC2, EKS, Fargate 또는 Lambda)를 선택합니다.

- 로드 테스트 프레임워크: 팀의 요구 사항에 맞게 사용 편의성, 성능, 유연성 및 커뮤니티 지원의 균형을 맞추는 로드 테스트 프레임워크(예: JMeter, Locust, Grafana K6 또는 Gatling)를 선택합니다.

시간 경과에 따른 최적화

GAMEOPS06: 시간이 지남에 따라 게임을 어떻게 최적화하나요?

주요 지표와 원격 측정 데이터를 모니터링하여 플레이어 추세, 시스템 성능 및 개선 영역을 식별하여 시간이 지남에 따라 게임을 최적화합니다. 이러한 인사이트를 기반으로 게임 설계, 인프라 및 로드 테스트 접근 방식을 지속적으로 업데이트하는 동시에 새로운 기술과 프레임워크에 적응하여 게임이 발전함에 따라 최적의 성능과 플레이어 경험을 제공합니다.

모범 사례

- GAMEOPS06-BP01 주요 게임 지표를 모니터링하여 플레이어 추세와 패턴을 식별하고 정보를 사용하여 게임을 개선합니다.
- GAMEOPS06-BP02 게임 변경에 따라 로드 테스트 접근 방식 업데이트 및 조정

GAMEOPS06-BP01 주요 게임 지표를 모니터링하여 플레이어 추세와 패턴을 식별하고 정보를 사용하여 게임을 개선합니다.

게임 클라이언트 시스템 사용량, 앱 사용량, 예외 및 충돌 데이터 외에도 게임 백엔드 시스템으로 전송되는 게임 원격 측정 데이터를 캡처합니다. 이 데이터는 플레이어가 게임의 다양한 기능과 상호 작용하는 방식을 이해할 수 있도록 플레이어 활동을 나타내야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

구현에 따라 게임 클라이언트는 사전 정의된 게임 기능 또는 게임 월드의 위치에서 원격 측정 데이터를 수집할 수 있습니다. 데이터는 처리를 위해 백엔드 수집 서비스로 전송됩니다. 백엔드 서비스에 연결할 수 없는 경우 클라이언트는 백엔드 서비스를 다시 사용할 수 있을 때까지 로컬 디바이스에 로컬로 데이터를 저장할 수 있습니다. 게임 디자이너는이 원격 측정 데이터를 사용하여 플레이어가 게임을 플레이하는 방식과 게임에 이상이 있는지 검토합니다.

예를 들어 플레이어 이동 및 맵의 항목과의 상호 작용을 원격 측정 데이터에서 추출하고 설정된 시간 동안 플레이어가 게임 내 활동의 히트 맵으로 표시할 수 있습니다. 이러한 데이터는 게임 디자이너가 무기의 성능, 게임 내 캐릭터의 성능 또는 맵의 복잡성과 같은 게임의 다양한 요소의 균형을 맞출 필요성을 식별하는 데 도움이 됩니다. 원시 원격 측정 데이터는 일반적으로 저장되었다가 처리되어 분석가가 시각화할 수 있는 분석을 추출합니다.

Game AnalyticsPipeline 솔루션 구현은 게임 개발자가 게임 및 서비스에서 생성된 원격 측정 데이터를 수집, 저장 및 분석하기 위해 확장 가능한 서버리스 데이터 파이프라인을 시작하는 데 도움이 됩니다. 이 솔루션은 데이터 스트리밍 수집을 지원하므로 사용자는 몇 분 안에 게임 및 기타 애플리케이션에서 인사이트를 얻을 수 있습니다.

사용자 지정 게임 원격 측정 데이터 수집, 저장, 처리 및 분석을 위해는 빅 데이터 처리 및 분석을 위한 다양한 전문 서비스 AWS 도 제공합니다.

구현 단계

- 게임 원격 측정 데이터 캡처: 플레이어 활동, 시스템 사용량, 예외 및 충돌에 대한 데이터를 수집하여 플레이어 상호 작용을 이해하고 문제를 식별합니다.
- 원격 측정 컬렉션 구현: 사전 정의된 게임 기능 또는 위치를 사용하여 원격 측정 데이터를 수집하여 백엔드 서비스에 전송하고 백엔드에 연결할 수 없는 경우 로컬에 저장합니다.
- AWS 분석 솔루션 사용: 게임 분석 파이프라인과 같은 AWS 서비스를 사용하여 확장 가능한 데이터 수집, 저장 및 분석과 특수 빅 데이터 처리 및 분석 서비스를 제공합니다.

GAMEOPS06-BP02 게임 변경에 따라 로드 테스트 접근 방식 업데이트 및 조정

로드 테스트 접근 방식을 최적화하는 것은 게임 개발 주기와 함께 발전해야 하는 지속적인 프로세스입니다. 게임의 복잡성, 사용자 기반 및 기능 세트가 증가함에 따라 로드 테스트 전략은 실제 조건을 정확하게 시뮬레이션하고 실행 가능한 인사이트를 제공하는지 확인하기 위해 조정되어야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

다음을 고려하세요.

누락되거나 오래된 테스트 시나리오

개발 프로세스 중에 게임에 새 기능이 추가되면 새 로드 테스트 시나리오를 생성하고 실행하여 새 기능의 성능과 확장성을 검증합니다. 마찬가지로 성능을 개선하거나, 플레이어 피드백을 처리하거나, 새로운 설계 목표에 맞게 기능을 리팩터링하는 경우가 많습니다. 따라서 변경 사항에 보조를 맞추고 시스템 상태를 실제로 테스트하고 반영하기 위해 테스트 시나리오를 지속적으로 업데이트해야 합니다.

새로운 로드 테스트 프레임워크

개발자는 다음과 같은 다양한 이유로 로드 테스트 프레임워크를 변경해야 할 수 있습니다.

- 초기 프레임워크는 더 이상 사용자 로드를 적절하게 시뮬레이션하거나 시스템 성능에 필요한 수준의 통찰력을 제공하지 못할 수 있습니다.
- 새로운 게임 기능을 사용하려면 새로운 프로토콜, APIs
- 개발자는 로드 테스트 프로세스에 더 익숙해지면 고급 기능을 원할 수 있습니다.
- 팀의 기술 전문 지식, 프로그래밍 언어 또는 기존 도구 체인에 더 잘 맞는 프레임워크 선호

시간이 지남에 따라 신중하게 평가하고 조정함으로써 개발자는 로드 테스트 프로세스를 게임의 변화하는 요구 사항에 맞추고 전체 사용자 경험을 최적화하고 개선하는 데 필요한 인사이트를 계속 제공할 수 있습니다.

비용 최적화

관리형 AWS 서비스 사용의 용이성과 편의성은 특히 개발 초기 단계에서 매우 유용할 수 있습니다. 이러한 서비스는 기본 인프라 관리를 추상화하여 팀이 솔루션을 빠르게 설정하고 로드 테스트 시나리오를 생성하고 결과를 분석하는 데에만 집중할 수 있도록 합니다. 그러나 관리형 서비스를 사용하면 인프라 프로비저닝, 구성 및 유지 관리와 고가용성, 규모 조정 및 모니터링 기능 제공과 같은 추가 가치와 편의성으로 인해 비용이 더 많이 들 수 있습니다.

팀이 성숙해지고 로드 테스트 프로세스에 더 익숙해지고 자신감을 가지게 되면 인프라를 자체 관리하면 추가 최적화와 비용 절감을 제공할 수 있는 경우가 발생할 수 있습니다. 이 실습 접근 방식은 운영 오버헤드를 높이지만 컴퓨팅 리소스, 구성, 조정 동작 및 리소스 사용률을 직접 제어하면 미세 조정 및 비용 절감을 위한 새로운 기회를 얻을 수 있습니다. 예를 들어 팀이 AWS Fargate 서버리스 아키텍처로 로드 테스트 여정을 시작한 다음 나중에 Amazon EKS 클러스터의 기본 노드를 자체 관리하는 것이 합리적일 수 있습니다.

구현 단계

- 테스트 시나리오 업데이트: 로드 테스트 시나리오를 지속적으로 생성 및 업데이트하여 새로운 기능과 리팩터링된 기능을 검증하고 게임의 현재 상태를 반영하는지 확인합니다.

- 로드 테스트 프레임워크 평가: 필요에 따라 새 프레임워크에 적응하여 사용자 로드를 시뮬레이션하고, 새 프로토콜을 지원하고, 팀의 전문 지식 및 도구 체인에 맞게 조정합니다.
- 비용 최적화: 쉽고 편리하게 관리형 AWS 서비스로 시작한 다음 팀이 로드 테스트 프로세스에 더 익숙해지면 비용 절감을 위해 자체 관리형 인프라를 고려하세요.

리소스

운영 우수성과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

설명서 및 블로그

- Game Tech의 아키텍처 모범 사례
- AWS pt.1에서 Game Studio 관리
- AWS pt에서 Game Studio 관리 2
- AWS pt에서 Game Studio 관리 3
- 모범 사례 AWS 환경 설정
- Control Tower 랜딩 존에 대한 다중 계정 전략
- 게임 분석 파이프라인
- 게임 분석 파이프라인을 사용하여 게임 데이터 인사이트 극대화
- 플레이어 인사이트를 위한 AWS Glue 및 Amazon Redshift Spectrum 활용
- 에서 CI/CD 파이프라인을 설정하는 방법
- AWS 빌드 파이프라인을 사용하여 Good Job Games가 43% 가속화하는 방법
- Unity 모바일 앱용 빌드 파이프라인 구현
- 기타 관련 CI/CD 블로그
- Game DevOps는 Game-Server CD Pipeline 블로그를 통해 쉽게 만들 수 있습니다.
- Harmony Games, 완전 사용자 지정 게임 백엔드 사용 AWS Cloud Development Kit (AWS CDK) (AWS CDK) 배포
- GameLift 시작 준비
- Amazon Gamelift Anywhere를 사용한 하이브리드 게임 서버 호스팅
- Amazon Gamelift Anywhere 및 Amazon Gamelift Agent를 사용하여 게임 서버 개발 속도 향상
- Amazon Gamelift를 사용하여 플레이어당 1 USD 미만으로 Unreal Engine 게임을 호스팅하는 방법
- 에서 확장 가능한 교차 플랫폼 게임 백엔드를 빌드하기 위한 새로운 솔루션 지침 AWS

- 에서 1백만 명의 동시 사용자에게 Pragma 백엔드 게임 엔진 로드 테스트 AWS
- Code Wizards가를 사용하여 200만 명의 동시 플레이어에게 헤로틱 랩의 나카마를 로드하는 방법 AWS
- 조직 단위 모범 사례
- AWS X-Ray
- AWS 카운트다운
- AWS 게임 솔루션 허브용

파트너 솔루션

- New Relic
- Splunk APM
- Backtrace.io
- Sentry
- Datadog APM
- Honeycomb.io

백서

- 여러 계정을 사용하여 환경 구성
- 에서 확장 가능한 게임 개발 패턴 소개 AWS

비디오

- YouTube 시리즈:에서 게임 빌드 AWS
- AWS 게임용: Boss LEVEL 팟캐스트
- Re:Invent 2023: 처음 1천만 명의 사용자에 AWS 대해 확장
- Re:Invent 2022: Riot Games가 매일 20TB의 분석을 처리하는 방법 AWS
- Re:Invent 2022: AWS 및 Riot Games가 거버넌스 보고 엔진을 구축한 방법
- Re:Invent 2023:에서 분산 설계 패턴 구현 AWS
- Re:Invent 2023: Mortal Kombat 1을 사용하여 멀티플레이어 게임을 수백만 개로 확장
- Re:Invent 2022: Netflix에서 카오스 엔지니어링의 진화

- Re:Invent 2023: 클라우드 거버넌스 모범 사례
- Re:Invent 2023:에서 다중 리전 아키텍처 생성 모범 사례 AWS

교육 자료

- 커리큘럼 - 게임 AWS 1부 시작하기

보안

보안 원칙에는 위험 평가 및 완화를 통해 비즈니스 가치를 제공하면서 정보, 시스템 및 자산을 보호하는 기능이 포함됩니다. 글로벌 가시성과 많은 수의 플레이어로 인해 게임은 악용자, 해커 및 시스템을 악용하고 침해하는 방법을 찾는 사람들에게 바람직한 대상입니다. 이로 인해 강력한 보안 기반이 마련되지 않은 경우 플레이어 경험이 실망하고 게임 개발자의 비용이 증가할 수 있습니다.

공동 책임 모델에 설명된 대로 보안의 어떤 측면이 고객의 책임 AWS 이고 어떤 측면이 고객의 책임인지 이해하여 강력한 보안 태세를 유지할 준비를 하는 것이 중요합니다. 이 원칙은 클라우드에서 게임을 개발하고 운영할 때 고려해야 할 모범 사례 클라우드 보안 지침을 제공합니다.

시스템을 설계하기 전에 액세스 제어를 포함하는 보안 모범 사례 세트를 설정해야 합니다. 또한 데이터 보호를 통해 데이터의 기밀성과 무결성을 유지하면서 보안 인시던트를 식별하고 시스템 및 서비스를 보호할 수 있어야 합니다. 보안 사고에 대응하기 위한 잘 정의된 프로세스를 마련하고 숙련해야 합니다. 이러한 도구와 기법은 재정적 손실 방지 또는 규제 의무 준수와 같은 비즈니스 목표를 지원하기 때문에 중요합니다.

고객 사례

AnyCompany Games는 보안 태세를 개선하는 가상 게임 스튜디오입니다. 보안은 직접 애플리케이션에 대한 설명이 있을 때 쉽게 이해할 수 있습니다. AnyCompany Games는이 섹션에서 원칙에 설명된 보안 모범 사례를 컨텍스트화하는 데 사용됩니다.

중점 영역

- 설계 원칙
- 보안 기초
- 지속적인 보안
- ID 및 액세스 관리
- 액세스 관리
- 탐지
- 인프라 보호
- 인시던트 대응
- 애플리케이션 보안
- 보안 자동화

- 위협 모델링
- 리소스

설계 원칙

Well-Architected Framework 백서의 보안 원칙에 따른 설계 원칙 외에도 다음 설계 원칙은 클라우드에서 게임 워크로드의 보안을 강화할 수 있습니다.

- 플레이어 사용 동작 모니터링 및 조정: 사용 데이터를 캡처하고 분석하여 플레이어가 게임 및 소셜 기능과 상호 작용하는 방식을 이해합니다. 이 데이터를 분석하면 플레이어 경험을 저하시킬 수 있는 모욕적이고 부적절한 행동을 감지하고 이에 대응할 수 있습니다.

보안 기초

GAMESEC01: 게임 개발을 위한 보안 기본 사항을 어떻게 구현하나요?

게임 스튜디오에는 개발 환경과 라이브 플레이어 서비스를 모두 보호하는 고유한 보안 접근 방식이 필요합니다. 게임 스튜디오를 위한 강력한 AWS 보안 전략에는 다중 계정 구조, 강력한 인증, IAM 정책을 사용한 명확한 권한 부여 전략이라는 세 가지 상호 연결된 구성 요소가 필요합니다. 다중 계정 AWS 구조를 사용하면 스튜디오에서 다양한 게임 프로젝트, 개발 단계 및 도구 환경을 분리할 수 있습니다. 이를 통해 스튜디오는 특정 환경 또는 서비스에 대한 액세스와 같은 사항을 보다 세밀하게 제어할 수 있습니다. 강력한 인증을 활성화하면 팀원이 스튜디오에서 작업하든 원격으로 작업하든 개발 리소스에 안전하게 액세스할 수 있는 동시에 소스 코드, 게임 빌드 및 독점 도구에 대한 엄격한 제어를 유지할 수 있습니다. 또한 Studio에는 IAM 권한 및 역할에 대한 최소 권한 원칙을 사용하여 권한을 부여하기 위한 명확한 권한 부여 전략이 있어야 합니다. IAM 역할을 사용하여 개발 팀에 하위 수준 AWS 서비스에 대한 액세스 권한을 부여하는 동시에 아티스트와 디자이너를 특정 자산 관리 및 빌드 시스템으로 제한하는 등 다양한 개발 팀 역할 간에 권한을 할당할 수 있습니다. 이 특수 접근 방식은 게임 스튜디오가 지적 재산을 보호하고, 효율적인 개발 워크플로를 유지하고, 팀을 안전하게 확장하는 동시에 개발자에게 프로젝트에서 빠르게 반복할 수 있는 적절한 액세스 권한을 제공할 수 있는지 확인합니다.

모범 사례

- GAMESEC01-BP01 계정 루트 사용자가 아닌 역할 및 페더레이션 액세스를 사용하여 AWS 환경에서 작업 수행
- GAMESEC01-BP02 AWS Control Tower 를 사용하여에서 다중 계정 환경을 빠르게 설정 AWS

- GAMESEC01-BP03 특정 직무에 맞게 조정된 최소 권한 역할 정책 사용
- GAMESEC01-BP04 역할 및 페더레이션 액세스 정책을 계정 수준 액세스 정책과 함께 사용하여 리소스에 AWS 대한 액세스 권한 부여
- GAMESEC01-BP05 중앙 ID 제공업체 사용

GAMESEC01-BP01 계정 루트 사용자가 아닌 역할 및 페더레이션 액세스를 사용하여 AWS 환경에서 작업 수행

를 처음 생성할 때 루트 사용자라는 자격 증명으로 AWS 계정시작합니다.이 자격 증명은 계정과 연결된 이메일 주소 및 암호를 사용하여 액세스합니다. 루트 사용자는 해당 계정 내의 AWS 서비스 및 리소스에 대한 전체 액세스 권한을 가집니다. 대부분의 경우 day-to-day 작업에 루트 사용자를 사용하지 않아야 합니다. 루트 수준 액세스가 필요한 경우 반드시 필요한지 확인하고 사용을 추적하기 위한 추가 로깅 및 가드레일이 있는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS Organizations 구성에서 각 계정에는 여전히 자체 루트 사용자가 있지만 대신 IAM 역할 및 IAM Identity Center 사용자를 통해 day-to-day 액세스를 관리해야 합니다. 게임의 수명 주기 단계 및 팀에 맞게 역할 기반 액세스를 생성합니다. 예를 들어 라이브 운영 팀은 게임 내 이벤트를 관리할 수 있는 권한이 필요한 반면 개발자는 업데이트를 푸시할 수 있는 액세스 권한이 필요할 수 있습니다. 타사 서비스 또는 파트너와 작업할 때는 페더레이션 액세스를 사용하여 민감한 인프라를 노출하지 않고 안전한 협업을 가능하게 합니다. 이 접근 방식은 게임 인프라 및 플레이어 데이터의 보안을 유지하면서 각 사용자 또는 파트너가 필요한 액세스 권한만 가지고 있는지 확인합니다.

고객 사례

AnyCompany Games는 새 게임을 개발할 때 역할 기반 액세스 제어를 구현했습니다. 다양한 개발 팀에 특정 IAM 역할을 사용하면 공유 자격 증명을 사용하지 않아도 됩니다. 이 설정을 사용하면 개발 팀이 핵심 게임 시스템의 역할을 수임할 수 있는 반면, 콘텐츠 팀의 역할은 자산 관리 서비스에만 액세스할 수 있습니다.

구현 단계

- 반드시 필요한 경우가 아니면 계정을 설정한 후에는 루트 사용자를 사용하지 마십시오. 계정을 생성하고 루트 사용자를 보호한 다음 필요한 관리 IAM 역할을 즉시 생성하고 페더레이션 사용자에게 해당 역할을 할당합니다.

- 루트 사용자만 사용할 수 있는 제한된 수의 작업을 수행해야 하는 경우에만 루트 사용자를 사용합니다. 이러한 작업의 예로는 루트 사용자 이메일 주소 변경 및 AWS 지원 플랜 변경이 있습니다.

GAMESEC01-BP02 AWS Control Tower 를 사용하여에서 다중 계정 환경을 빠르게 설정 AWS

단일 계정으로만 AWS 를 사용하기 시작하면 게임 개발 프로세스가 진행됨에 따라 게임 스튜디오가 확장될 수 있습니다. 예를 들어 단일를 사용하면 서비스 한도에 도달하기 AWS 계정시작하거나 다양한 프로젝트 및 워크로드에 대한 비용이 더 복잡해질 수 있습니다. 게임 제목과 환경이 서로 다른 계정을 생성하면 팀이 새로운 기능을 실험하고, 서비스 제한을 우회하고, 보안 태세와 규정 준수를 유지할 수 있습니다. 에서 다중 계정 전략을 구현하면 여러 계정에 서비스 제한을 분산하고 AWS 비용에 대한 인사이트를 얻을 AWS수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

여러를 사용하면 AWS 계정 자동으로 더 혼란스럽고 시간이 많이 걸린다는 일반적인 오해입니다. 대신 여러 계정의 거버넌스를 촉진하도록 설계된 AWS 서비스를 사용하면 게임 스튜디오가 계정 관리에 소요되는 시간을 줄일 수 있습니다.

AWS Control Tower 를 사용하여 다중 계정 AWS 환경을 안전하게 프로비저닝할 수 있습니다. Control Tower는 새 AWS 환경을 구축하거나, 여정을 시작하거나 AWS, 완전히 새로운 경우에 권장됩니다 AWS. 짧은 설정 프로세스 중에 AWS Organizations Service Catalog 및 AWS IAM Identity Center와 같은 계정 및 사용자 액세스 관리와 관련된 다른 AWS 서비스와 통합할 수 있습니다.

고객 사례

AnyCompany Games는 처음에 단일에서 운영되었으며 AWS 계정중요한 베타 테스트 중에 게임 개발 팀 중 하나가 EC2 서비스 제한에 도달하면 여러 장애물에 도달합니다. 동시에 다른 게임의 개발 팀은 자동화된 테스트 파이프라인에 대한 리소스 할당에 어려움을 겪었습니다. 이 상황은 AnyCompany Games가 프로젝트 간에 비용을 정확하게 분리할 수 없어 각 게임의 개발에 대한 예산을 책정하기 어려운 중단점에 도달했습니다.

그런 다음 AnyCompany Games는를 사용하여 다중 계정 전략을 구현했습니다 AWS Control Tower. 이들은 고유한 개발, QA 및 프로덕션 환경을 사용하여 각 게임 프로젝트에 대해 별도의 계정을 생성했습니다. 이 계정 수준 분리는 각 프로젝트 데이터와 자산을 격리하므로 한 게임에서 작업하는 팀은 다른 게임에서 리소스에 액세스하거나 수정할 수 없습니다. AWS Organizations이를 통해 각 게임의 인

프라 비용을 명확하게 보여주는 중앙 집중식 결제 구조를 수립하고 조직 전체의 액세스 정책을 만들었습니다.

구현 단계

- AWS Control Tower를 사용하여 자동화된 다중 계정 환경을 설정합니다.
- 환경(예: 개발, QA 및 프로덕션)을 기반으로 계정을 구성합니다.
- AWS IAM Identity Center 및 Service Catalog를 사용하여 사용자 권한을 중앙 집중화하고 계정 간 리소스 프로비저닝을 간소화합니다.

GAMESEC01-BP03 특정 직무에 맞게 조정된 최소 권한 역할 정책 사용

IAM 정책 구성은 강력한 보안 기반을 구축하는 데 필수적인 부분입니다. IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 예를 들어 QA 팀은 테스트 환경에서 사물을 변경할 수 있는 액세스 권한이 필요하지만 프로덕션 환경을 수정할 수는 없어야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

워크로드 또는 사용 사례에 필요한 권한을 탐색하는 동안 관리형 정책과 같은 광범위한 권한으로 시작할 수 있습니다. 사용 사례가 발전함에 따라 최소 권한을 향해 나아가도록 부여하는 권한을 줄이기 위해 노력할 수 있습니다.

구현 단계

- 사용자 및 애플리케이션에 대한 IAM 역할을 생성하기 위한 최소 권한 관행을 따릅니다.
- 사용 AWS관리형 정책은 팀 또는 애플리케이션이 작업을 수행하는 데 필요한 특정 권한을 식별하면서 광범위한 액세스를 신속하게 제공합니다.
- 또한 Studio는 IAM 액세스 분석기 정책 생성을 사용하여 IAM 항목에서 사용하는 작업 및 서비스를 식별하는 CloudTrail 이벤트를 기반으로 사용자 지정 IAM 정책을 생성할 수 있습니다.
- IAM 정책을 정기적으로 검토하고 지나치게 허용적인 정책을 편집합니다.

GAMESEC01-BP04 역할 및 페더레이션 액세스 정책을 계정 수준 액세스 정책과 함께 사용하여 리소스에 AWS 대한 액세스 권한 부여

신규 AWS 사용자는 다른 사용자에게 액세스 권한을 부여할 때만 IAM 정책을 사용하는 경우가 많습니다. 그러나를 사용하는 경우 서비스 제어 정책을 IAM 정책과 함께 사용하여 스튜디오 팀원과 계약자에게 필요한 수준의 액세스 권한을 부여하는 방법을 AWS Organizations고려하세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

IAM 정책을 생성하여 작업하는 서비스 또는 API 작업에 대한 AWS 액세스를 허용하거나 거부할 수 있습니다 AWS Identity and Access Management. 사용자, 그룹 또는 역할과 같은 IAM 자격 증명에만 적용할 수 있습니다. 예를 들어 IAM 정책을 사용하여 사용자에게 Amazon S3에 대한 읽기 전용 액세스 권한을 제공할 수 있습니다.

서비스 제어 정책(SCPs)은의 가드레일입니다 AWS 계정. SCP는 권한을 부여하지 않으며 개별 멤버 계정의 AWS 서비스에 대한 작업을 제한하는 데 사용됩니다. 예를 들어 SCP는의 특정 리전 AWS 계정 액세스를 거부할 수 있습니다.

작업이 수행되면 관련 IAM 정책이 SCPs와 함께 평가됩니다. 이전 예제의 후속 작업은 역할이 EC2 인스턴스를 실행하려는 시도이고, IAM은 EC2 인스턴스가 허용되는지(ec2:RunInstances의 경우 "허용")를 나타내며, SCPs는 선택한 리전이 유효한지 여부를 결정합니다("us-east-1"은 허용되지만 "us-west-1"은 SCP에 의해 거부됨).

IAM 정책 및 SCPs를 계층화하면 AWS 리소스에 액세스하는 모든 사용자에게 필요한 적절한 권한만 부여되는지 확인할 수 있습니다. 이는 AWS 계정 및 리소스가 여러 리전에 걸쳐 있지만 게임 스튜디오 내의 모든 사용자가 모든 리전에 액세스해야 하는 것은 아닌 경우 특히 중요합니다.

특정 팀에 게임 구성 업데이트, 플레이어 데이터 관리, 프로모션 이벤트 구성, 사용자 생성 콘텐츠 조정과 같은 특정 권한을 부여하도록 IAM 정책을 조정할 수 있습니다. 한편 SCPs 사용하여 게임 운영에 중요한 조직 전체의 제어를 적용할 수 있습니다. 여기에는 게임이 작동하는 승인된 리전으로만 배포를 제한하고, 민감한 플레이어 데이터 스토어에 대한 무단 액세스를 방지하고, 규정 준수 요구 사항을 적용하고, 개발 계정 전체에서 서비스 사용을 제한하여 비용을 제어하는 것이 포함될 수 있습니다.

구현 단계

- IAM 정책을 사용하여 개별 사용자, 그룹 또는 역할에 대한 권한을 관리합니다.
- 에서 서비스 제어 정책(SCPs) AWS Organizations 을 사용하여 계정 수준 권한을 적용합니다.
- IAM 정책과 SCPs 결합하여 특정 사용자 및 계정에 필요한 액세스 권한만 부여합니다.

리소스

- AWS IAM의 정책 및 권한
- 서비스 제어 정책
- 직무에 관한AWS 관리형 정책

GAMESEC01-BP05 중앙 ID 제공업체 사용

중앙 자격 증명 공급자는 사용자 자격 증명, 자격 증명, 권한 및 인증을 저장하고 관리하기 위한 단일 소스 역할을 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

중앙 ID 제공업체를 사용하여 사용자 인증 프로세스를 간소화하고, 일관된 보안 정책을 적용하고, AWS 계정 및 애플리케이션 전반에서 사용자 관리를 간소화할 수 있습니다. 중앙 집중식 접근 방식을 사용하면 사용자 ID와 자격 증명을 별도로 관리할 필요가 없으므로 불일치, 중복 및 기타 보안 취약성의 위험이 줄어듭니다. 사용자 자격 증명과 인증을 한 곳에 통합하면 전체 AWS 환경에 대한 가시성, 제어 및 감사 가능성도 향상됩니다.

고객 사례

AnyCompany Games는 빠르게 확장 AWS 되는 인프라에서 개발자 액세스를 관리하는 데 상당한 어려움을 겪었습니다. 개발 팀은 세 가지 주요 타이틀에서 50명에서 200명으로 성장했습니다. 처음에는 각 프로젝트 팀이 자체 AWS 액세스 자격 증명을 관리하여 보안 관행이 일관되지 않고, 신규 개발자의 온보딩이 지연되고, 가끔 보안 인시던트가 발생합니다.

스튜디오는 AWS IAM Identity Center를 중앙 ID 제공업체로 구현하여 사용자 관리를 단일 시스템으로 통합했습니다. 개발자가 동일한 회사 자격 증명을 사용하여 액세스할 수 있도록 기존 회사 디렉터리와 AWS 연결했습니다. 이제 개발자는 기존의 단일 회사 로그인을 사용하여 작업을 완료하는 데 필요한 AWS 액세스 권한을 얻습니다.

구현 단계

- AWS IAM Identity Center를 중앙 ID 제공업체로 사용하는 것이 좋습니다. 이를 통해에서 일관된 액세스 관리를 제공하고 AWS 계정, 직원에게 Single Sign-On 인증을 제공하며, AWS 애플리케이션에 대한 사용자 액세스 감사를 간소화할 수 있습니다. 또한 IAM Identity Center는 지원되는 자격 증명 공급자의 기존 기업 자격 증명과 연결됩니다.

지속적인 보안

GAMESEC02: 지속적인 보안 모범 사례를 어떻게 달성, 유지 및 모니터링하나요?

업계 전반의 기업, 특히 게임 업계에서는 모범 보안 사례를 준수하는 것이 가장 중요합니다. 게임 산업은 플레이어 신뢰를 구축하고 유지하며 강력한 평판을 얻는 데 의존하며, 사소한 보안 문제라도 이러한 신뢰도를 빠르게 약화시킬 수 있습니다.

또한 게임 산업의 글로벌 특성상 게임이 제공되는 리전에서 데이터 보호, 소비자 개인 정보 보호 및 보안을 관리하는 다양한 산업 규정 및 표준을 준수해야 합니다. 공정하고 안전한 게임 플레이는 강력한 보안 조치의 중요성을 강조하는 또 다른 중요한 측면입니다. 사기, 해킹 및 기타 형태의 게임 악용은 합법적인 플레이어의 게임 경험을 방해할 수 있으므로 게임 플레이의 무결성을 유지하고 참가자를 위한 레벨 플레이 영역을 조성하는 데 강력한 보안 제어가 필수적입니다.

모범 사례

- GAMESEC02-BP01 표준 보안 관행을 위한 즉시 배포 가능 템플릿 사용
- GAMESEC02-BP02 보안 이벤트가 발생할 때 자동 문제 해결 기법 사용

GAMESEC02-BP01 표준 보안 관행을 위한 즉시 배포 가능 템플릿 사용

Ready-to-deploy 가능한 템플릿은 클라우드에서 보안 태세를 평가하는 선제적이고 민첩한 방법을 제공합니다. 사전 구성된 템플릿은 클라우드 보안을 평가하고 필요한 변경 사항을 즉시 구현합니다. 템플릿은 다양한 기술과 널리 허용되는 보안 프레임워크에 걸친 다양한 모범 사례를 포함합니다. 템플릿을 사용하면 게임 스튜디오가 일관된 인프라 구성을 유지하는 데 도움이 될 수 있습니다. 특히 새로운 워크로드를 AWS 계정 지원하기 위해 확장하고 추가할 수 있기 때문입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

AWS 서비스를 사용하고 ready-to-deploy 템플릿을 구현하면 게임 개발자는 정기적인 보안 평가 및 지속적인 모니터링을 통해 클라우드 보안 태세를 사전에 평가하고 강화하며, 지적 재산을 보호하고, 플레이어 데이터를 보호하고, 안전한 게임 환경을 조성하여 잠재적 취약성을 즉시 식별하고 해결할 수 있습니다.

고객 사례

AnyCompany Games는 유럽 산업에서 다음 타이틀을 출시할 준비를 할 때 상당한 문제에 직면했습니다. 기존 데이터 처리 관행이 GDPR 요구 사항을 충족하지 않는다는 것을 깨달았습니다. 솔루션에 대해 AWS Security Hub CSPM 및 AWS Config 와 ready-to-deploy 가능한 템플릿을 사용했습니다. 팀은 GDPR 표준과 비교하여 기존 인프라를 AWS Config자동으로 평가한 GDPR별 적합성 팩을에 구현했습니다. 이 초기 스캔에서는 플레이어 데이터가 저장된 위치에 대한 부적절한 데이터 보존 정책 및 부적절한 액세스 제어와 같은 몇 가지 중요한 격차가 드러났습니다. 템플릿의 사전 정의된 규칙을 사용하여 AnyCompany Games는 필요한 변경 사항을 신속하게 구현했습니다. 또한 템플릿에서 제공하는 지속적인 자동 규정 준수 검사를 통해 소규모 팀은 게임을 계속 업데이트하고 확장하더라도 GDPR 규정 준수를 손쉽게 유지할 수 있었습니다.

구현 단계

- 의 관리형 규칙 및 적합성 팩과의 표준과 같은 AWS Config 표준 보안 사례에 템플릿을 사용합니다 AWS Security Hub CSPM.
- Security Hub CSPM 표준의 세부 정보를 검토하여 게임 스튜디오의 보안 요구 사항에 가장 적합한 표준을 결정합니다.

GAMESEC02-BP02 보안 이벤트가 발생할 때 자동 문제 해결 기법 사용

게임 개발자는 자동화된 문제 해결 기술을 사용하여 게임 인프라를 사전에 보호 및 유지 관리하고 보안 인시던트가 미칠 수 있는 잠재적 영향을 최소화할 수 있습니다. 보안 문제가 감지되면 실행서를 사용하여 상황에 대한 대응을 안내합니다. 가능한 경우 이러한 응답을 자동화하여 문제를 더 빠르게 해결하고 영향을 줄입니다. 이렇게 하면 게임 가동 중지 및 중단 가능성을 줄여 플레이어 경험이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

보안 문제에 대응할 준비를 하면 플레이어의 경험을 보호할 뿐만 아니라 다양한 규정 준수 및 규제 표준을 충족할 수 있습니다. 또한 자동화된 보안 응답을 사용하면 워크로드가 확장됨에 따라 보안 작업이 확장됩니다.는 이러한 인시던트에 대한 대응을 식별하고 자동화하는 데 도움이 되는 서비스를 AWS 제공합니다.

고객 사례

AnyCompany Games는 정기 자산 파이프라인 업데이트 중에 출시되지 않은 캐릭터 모델 및 텍스처가 포함된 S3 버킷이 실수로 공개되었을 때 심각한 보안 사고에 직면했습니다. 자동 보안 시스템이 수정 후 몇 분 이내에 버킷 권한 변경을 감지했습니다. 시스템은 버킷을 프라이빗 상태로 되돌리기, 노출 기

간 동안 액세스 시도 로깅, 보안 팀에 알림, 권한 변경에 대한 자세한 CloudTrail 로그 생성 등 문제 해결 실행서를 즉시 실행했습니다.

구현 단계

- 솔루션의 자동 보안 대응 AWS을 사용하여 보안 이벤트에 대한 응답으로 자동으로 수행할 작업을 정의하는 자동화 실행서를 구현합니다 AWS Security Hub CSPM.

리소스

- AWS for Games 블로그 -에서 Game Studio 관리 AWS: 1부
- AWS for Games 블로그 - 2 AWS 부에서 Game Studio 관리
- 에 기존 조직 단위 등록 AWS Control Tower
- AWS 계정 루트 사용자
- 루트 사용자 자격 증명이 필요한 작업
- 의 자동 보안 응답 AWS

ID 및 액세스 관리

GAMESEC03: 플레이어 자격 증명 및 액세스 관리를 어떻게 관리하나요?

게임을 개발할 때 플레이어에게 게임 및 관련 서비스에 대한 액세스 권한을 제공하는 방법을 결정해야 합니다. 다음 섹션에서는 플레이어 인증, 권한 부여 및 다중 인증 등 설계 고려 사항을 살펴봅니다.

모범 사례

- GAMESEC03-BP01 게임 환경 및 리소스에 대한 플레이어 액세스를 식별하고 제어하는 접근 방식 결정
- 게임 백엔드 서비스로 전송되는 GAMESEC03-BP02 인증 요청
- GAMESEC03-BP03 게임 백엔드 서비스를 사용하여 멀티플레이어 게임에 참가하기 위한 플레이어 요청 검증
- GAMESEC03-BP04 강력한 암호를 요구하여 플레이어 사용자 계정에 엄격한 보안 정책 적용
- GAMESEC03-BP05 플레이어가 계정에 다중 인증(MFA)을 설정할 수 있는 옵션을 제공합니다.

GAMESEC03-BP01 게임 환경 및 리소스에 대한 플레이어 액세스를 식별하고 제어하는 접근 방식 결정

이 결정은 플레이어 확보 및 수익화 전략, 플레이어 경험, 게임 게시 파트너가 제공할 수 있는 기존 기능과 같은 기타 요인의 영향을 받습니다. 예를 들어 게임에는 구매가 필요하며 플레이어가 실제 결제 방법을 자신의 계정과 연결하기 위해 사용자 프로필을 생성해야 할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

또는 게임을 플레이하기 전에 사용자 계정을 생성할 필요가 없어 플레이어가 처음으로 게임을 시도할 가능성이 높아짐으로써 게임이 최초 플레이어 경험의 진입 장벽을 줄이고 싶을 수 있습니다. 일반적으로 게임은 게임에 대한 플레이어 자격 증명 및 액세스 관리 접근 방식을 하나 이상 조합하여 구현합니다.

미인증 또는 익명 액세스

이 액세스 수준은 게임에서 플레이어가 소셜 네트워크 및 게임 시스템에서 새 사용자 계정을 생성하거나 자격 증명과 연결할 필요가 없는 경우에 유용합니다. 이는 플레이어가 게임 플레이를 시작하는 가장 간단하고 빠른 방법이며 게임 개발자가 초기 경험의 진입 장벽을 줄이고 싶을 수 있는 모바일 게임에서 특히 유용합니다.

이 액세스 시나리오에서는 게임 설치에서 사용량을 식별하려는 경우 플레이어의 디바이스에 고유 식별자를 생성하고 저장하도록 게임 클라이언트를 프로그래밍할 수 있습니다. 이 고유 식별자는 디바이스의 게임 세션에서 플레이어를 식별하고 시간 경과에 따른 사용량에 대한 분석 보고를 허용하는 데 사용됩니다. 나중에 플레이어가 계정을 생성하도록 선택한 경우 새 사용자 계정을 이전에 생성된 고유 식별자와 연결할 수 있습니다. 이렇게 하면 새 플레이어 자격 증명이 과거 사용량과 연결되며, 여기에는 통계 및 게임 업적이 포함될 수 있습니다.

플레이어가 결국 계정을 생성하고 연결하지 않으면 플레이어가 게임과 상호 작용하는 데 사용하는 디바이스를 고유하게 식별할 수 있지만 플레이어에 대한 복구 가능한 정보는 수집 및 저장되지 않습니다. 따라서 플레이어가 디바이스를 끊기거나 분실하면 디바이스와 연결된 이전 저장 데이터도 손실되고 복구할 수 없을 수 있습니다.

사용자 이름 및 암호를 사용한 인증

게임은 플레이어가 게임의 백엔드 내에 저장된 사용자 이름과 암호로 자신의 사용자 계정을 생성하도록 허용할 수 있습니다. 이는 게임 개발자가 이미 개발자가 통합할 수 있는 기존 플레이어 계정 시스템

이 있는 게임 게시자와 협업할 때 발생할 수 있습니다. 또는 자체 게임을 게시하는 개발자는 플레이어가 게시하는 게임에서 액세스할 수 있는 단일 사용자 계정을 만들 수 있도록 하여 플레이어 경험을 간소화할 수 있습니다.

타사 소셜 네트워크 및 게임 시스템과의 인증 및 계정 연결

온라인 게임과 소셜 기능이 있는 게임은 플레이어 경험을 간소화하기 위해 타사 자격 증명 공급자 페더레이션을 제공하는 것이 일반적입니다. 플레이어에게 인증을 위해 사용자 이름과 암호 조합을 생성하도록 요청하는 대신 자격 증명 연동을 사용하여 플레이어가 소셜 네트워크 및 게임 시스템에서 타사 계정을 사용하여 인증할 수 있도록 할 수 있습니다. 이 로그인 프로세스는 플레이어의 로그인 및 등록 경험을 간소화합니다. 또한 필수 계정 생성에 대한 편리한 대안과 플레이어가 게임에 액세스할 수 있는 원활한 방법을 제공합니다.

게임 개발자에게 페더레이션 로그인 프로세스는 간소화된 플레이어 확인 워크플로를 제공할 수 있습니다. 또한 개인화에 사용되는 플레이어 데이터를 관리하는 보다 신뢰할 수 있는 방법을 제공할 수 있습니다. 이는 플레이어에게 이미 타사 자격 증명 공급자에게 제공했을 가능성이 있는 특정 데이터를 제공하도록 요청할 필요가 없기 때문입니다. 또한 이러한 시스템은 플레이어를 친구와 연결하는 기능과 같은 추가 소셜 기능과의 통합을 제공합니다.

구현 단계

- 인증되지 않은 액세스 또는 익명 액세스를 사용하면 고유한 디바이스 식별자를 생성하여 사용량을 추적하고 나중에 계정 연결을 활성화하여 최초 플레이어의 장벽을 줄일 수 있습니다.
- 기존 플레이어 계정 시스템을 사용하거나 게임 전반에서 통합된 경험을 생성하여 전용 사용자 계정에 대한 사용자 이름 및 암호 인증을 구현합니다.
- 페더레이션 인증을 위해 타사 자격 증명 공급자를 통합하여 로그인 프로세스를 간소화하고 소셜 기능 및 개인화 데이터에 대한 액세스를 활성화합니다.

게임 백엔드 서비스로 전송되는 GAMESEC03-BP02 인증 요청

게임 백엔드 서비스로 전송되는 요청을 인증하면 원치 않는 요청이 성공하지 못할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

플레이어가 로그인할 수 있도록 인증 서비스를 제공해야 합니다.이 서비스는 플레이어가 성공적으로 인증할 때 JSON 웹 토큰(JWT)과 같은 수명이 짧은 보안 토큰을 게임 클라이언트에 반환해야 합니다.

이러한 토큰에는 플레이어 속성 및 기타 관련 메타데이터가 포함된 클레임 어설션이 포함될 수 있습니다. 이 관련 메타데이터는 게임 클라이언트에서 게임 백엔드로 전송되는 후속 요청에 사용하여 요청을 인증하고 인증된 플레이어의 컨텍스트에서 권한을 부여할 수 있습니다.

지속적인 개선 및 유지 관리가 필요한 자체 플레이어 인증 시스템을 설계 및 빌드하거나 Amazon Cognito에서 제공하는 확장 가능하고 안전한 사용자 가입, 로그인 및 액세스 제어 기능을 사용할 수 있습니다.

Amazon Cognito 사용자 풀에는 인증 및 권한 부여를 위한 사용자 디렉터리가 포함되어 있습니다. 사용자 풀은 타사 자격 증명 공급자와 통합할 수 있는 가입, 로그인 및 암호 재설정 워크플로를 위해 게임에 통합할 수 있는 APIs를 제공합니다. Application Load Balancer 및 Amazon API Gateway는 모두 Cognito와의 통합을 제공하여 이러한 서비스로 호스팅되는 사용자 지정 게임 백엔드로 전송된 요청에 대한 사용자 인증을 통합합니다.

게임이 익명 액세스를 지원하고 플레이어를 인증할 수 없는 경우 클라이언트 인증 접근 방식을 사용하여 게임 백엔드와 통합할 때 더 안전한 환경을 제공할 수 있습니다. 게임 클라이언트가 AWS 서비스를 직접 사용하는 경우 자격 증명을 사용하여 이러한 서비스에 대한 요청에 서명해야 합니다. 인증되지 않은 사용자를 위해 게임 클라이언트에 자격 증명을 제공하려면 SDK를 AWS 사용하여 AWS 서비스에 대한 요청에 서명하는 데 사용할 수 있는 Amazon Cognito 자격 증명 풀에서 수명이 짧은 자격 증명을 검색할 수 있습니다. 이러한 자격 증명은 게임 클라이언트에서 새로 고칠 수 있습니다.

게임 클라이언트의 AWS SDK와 직접 통합하는 것 외에도 사용자 지정 권한 부여를 지원하는 Amazon API Gateway와 같은 서비스를 사용하여 자체 게임 백엔드를 구축할 수도 있습니다. 자체 게임 백엔드 서비스를 설계하면 사용자 지정 서버 측 로직을 사용하여 요청을 신뢰할 수 있는 방식으로 제어할 수 있습니다.

Amazon GameLift를 사용하여 호스팅되는 게임을 위한 백엔드 서비스를 구축하는 방법에 대한 자세한 내용은 게임 클라이언트 서비스 설계를 참조하세요.

고객 사례

AnyCompany Games는 관리형 인증 및 권한 부여 접근 방식을 채택하여 다음 제목의 보안을 강화했습니다. 사용자 지정 사용자 이름 및 암호 시스템을 유지 관리하는 대신 Amazon Cognito 사용자 풀을 사용하여 플레이어 가입 및 로그인을 처리하고 자격 증명 풀을 사용하여 계정을 생성하기 전에 훈련 모드를 시도하는 플레이어의 익명 액세스를 지원했습니다. 또한 Cognito에 정의된 관리자 역할을 인식하여 해당 사용자에게 게임 내 특수 관리 기능에 대한 액세스 권한을 부여하는 게임 내에 사용자 지정 권한 부여 로직을 구현했습니다.

구현 단계

- Amazon Cognito 사용자 풀을 사용하여 JWTs와 같은 보안 토큰으로 인증을 관리하여 가입, 로그인 및 암호 재설정과 같은 기능을 활성화합니다.
- 익명 사용자가 AWS 서비스와 안전하게 상호 작용할 수 있도록 Amazon Cognito 자격 증명 풀에서 수명이 짧은 자격 증명을 검색합니다.
- 사용자 지정 서버 측 인증 로직을 위해 Amazon API Gateway를 사용하여 사용자 지정 게임 백엔드를 구현합니다.

GAMESEC03-BP03 게임 백엔드 서비스를 사용하여 멀티플레이어 게임에 참가하기 위한 플레이어 요청 검증

일반적으로 멀티플레이어 게임에서 플레이어는 사용 가능한 세션 목록에서 직접 옵션을 선택하여 게임 세션에 참여하거나 매치를 찾기 위한 요청을 제출합니다. 후자의 접근 방식은 게임 개발자가 적합한 게임 세션을 찾고 연결 정보(일반적으로 IP 주소 및 포트 번호)를 플레이어의 게임 클라이언트에 다시 제공할 책임이 있습니다. 구현은 개발 중인 게임의 장르에 따라 다를 수 있지만, 플레이어의 게임 참여 요청에 대한 서버 측 검증을 수행하는 것이 보안 모범 사례입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

예를 들어 세션 기반 멀티플레이어 게임에서 플레이어의 게임 세션 참여 요청은 서버에 대한 연결을 승인하기 전에 게임 서버 소프트웨어가 게임 백엔드 매치메이킹 서비스를 통해 검증해야 합니다. 플레이어가 게임 세션에 참가하도록 요청하면 게임 서버는 이전에 게임 백엔드 매치메이킹 서비스에서 게임 클라이언트에 제공한 플레이어 세션 ID 및 서버 생성 티켓과 같은 고유 식별자에 대한 요청을 확인해야 합니다.

게임 서버에 대한 연결을 시작하면 서버 측 소프트웨어가이 정보를 사용하여 매치메이킹 서비스에 플레이어의 연결 요청이 유효한지 확인하고 플레이어가 이전에 게임 세션에서 다른 플레이어에게 예약된 스팟에 조인하지 않는지 확인할 수 있습니다.

Amazon GameLift에서 호스팅되는 게임의 경우 이러한 유형의 서버 측 검증을 구현할 수 있는 방법에 대한 예는 Amazon GameLift 서버와의 게임 클라이언트/서버 상호 작용을 참조하세요.

고객 사례

AnyCompany Games의 초기 베타 출시 중에 플레이어가 게임 서버에 직접 연결하여 매치메이킹 시스템을 우회하여 심각한 경쟁 무결성 문제가 발생하고 있음을 발견했습니다. 순위가 높은 플레이어가 서

버 IP 주소를 친구와 공유할 수 있음을 발견했을 때 스킬 기반 매치메이킹 시스템을 우회하기 시작했으며, 그 결과 경험이 많은 플레이어가 초보 매치에 참여하고 새 플레이어에게 좌절감을 주는 경험을 하게 되었습니다. AnyCompany Games는 각 매치메이킹 요청에 대해 고유한 세션 티켓을 생성한 서버 측 검증 시스템을 구현하여 응답했습니다. 시스템에서는 플레이어 IDs와 매치메이킹 요청 티켓이 모두 필요하고 백엔드 매치메이킹 서비스에 대한 연결 시도가 확인되었습니다.

구현 단계

- 플레이어 세션 IDs 및 서버 생성 티켓과 같은 고유 식별자를 사용하여 플레이어 조인 요청을 서버 측에서 검증합니다.
- 매치메이킹 서비스와의 연결 요청 유효성을 확인하여 무단 액세스를 차단합니다.
- 검증 프로세스 중에 권한이 없는 플레이어가 게임 세션의 예약 스팟에 액세스하지 않는지 확인합니다.

GAMESEC03-BP04 강력한 암호를 요구하여 플레이어 사용자 계정에 엄격한 보안 정책 적용

게임이 플레이어에게 암호로 사용자 계정을 생성할 수 있는 기능을 제공하는 경우 강력한 정책을 준수하기 위해 플레이어의 암호가 필요합니다. 예를 들어 Amazon Cognito 사용자 풀은 사용자 계정에 대한 암호 요구 사항을 정의할 수 있는 기능을 제공합니다. 강력한 암호 정책을 설정하면 플레이어의 계정이 소셜 엔지니어링 및 무차별 대입 공격을 통해 추월되지 않도록 보호할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

고객 사례

AnyCompany Games는 약한 암호 정책으로 인해 인기 있는 타이틀에 계정 하이재킹 파도가 발생했을 때 긴급 상황에 직면했습니다. "password123"과 같은 간단한 암호를 사용하는 플레이어는 자동 무차별 대입 공격의 피해자가 되어 항목이 손실되고 게임 내 통화가 손상되었습니다. 이를 해결하기 위해 AnyCompany Games는 로그인 시스템을 개량하고 이전에 암호를 사용하지 않도록 규정했으며, 대문자, 숫자, 특수 문자, 최소 15자를 포함해야 합니다.

구현 단계

- 보안을 강화하려면 플레이어 계정에 강력한 암호 정책이 필요합니다.
- Amazon Cognito 사용자 풀을 사용하여 암호 요구 사항을 정의하고 적용합니다.

GAMESEC03-BP05 플레이어가 계정에 다중 인증(MFA)을 설정할 수 있는 옵션을 제공합니다.

플레이어 계정은 특히 게임 내 통화 및 구매를 지원하는 게임에서 악의적인 행위자의 자산이 될 수 있습니다. 플레이어 계정 해킹 및 소셜 엔지니어링 공격의 만연성으로 인해 플레이어에게 멀티 팩터 인증(MFA)을 구성하여 계정의 보안을 강화할 수 있는 옵션을 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

플레이어가 MFA를 사용하여 계정에 액세스하려고 하면 이메일 주소, 전화번호 또는 특별히 구축된 멀티 팩터 인증 모바일 앱으로 임시 코드가 전송됩니다. 성공적으로 인증하려면 플레이어가 제한된 기간 내에 로그인 시스템에 코드를 입력해야 합니다.

MFA는 새 지리적 위치에서 인증을 시도하는 계정, 플레이어 지원으로 플래그가 지정된 계정, 심지어 장기간 게임에 로그인하지 않은 계정을 보호하는 데에도 사용할 수 있습니다.

예를 들어 Amazon Cognito 사용자 풀은 사용자 디렉터리에서 다중 인증을 구성할 수 있습니다.

구현 단계

- 다중 인증(MFA)을 활성화하여 플레이어 계정 보안을 강화합니다.
- 이메일, 전화 또는 MFA 앱을 통해 전송된 임시 코드를 사용하여 계정 액세스를 확인합니다.
- 새 지리적 위치, 플래그가 지정된 계정 또는 장시간 활동이 없는 계정에 MFA를 적용합니다.

액세스 관리

GAMESEC04: 게임 콘텐츠에 대한 무단 액세스를 차단하려면 어떻게 해야 하나요?

최신 게임에는 플레이어 참여 및 게임 수익화의 중요한 측면인 다운로드 가능한 콘텐츠(DLC)와 같은 상당한 양의 콘텐츠가 포함됩니다. 플레이어는 새로운 캐릭터, 레벨 및 챌린지의 지속적인 스트림을 기대하므로 게임 개발자는 플레이어를 유지하기 위해 새로운 콘텐츠에 대한 지속적인 수요를 따라잡아야 합니다. 콘텐츠의 다양성과 크기는 게임 유형과 게임이 재생되는 디바이스에 따라 크게 달라질 수 있습니다. 게임 시스템에 관계없이 무단 액세스로부터 게임 콘텐츠를 보호합니다.

모범 사례

- GAMESEC04-BP01 권한 있는 클라이언트 및 사용자로 다운로드 가능한 콘텐츠에 대한 액세스 제한
- GAMESEC04-BP02 승인된 콘텐츠 전송 네트워크(CDNs.
- GAMESEC04-BP03 무단 액세스를 제한하는 지리적 제한 구현
- GAMESEC04-BP04 디지털 권한 관리(DRM) 솔루션을 사용하여 콘텐츠에 대한 액세스 제한

GAMESEC04-BP01 권한 있는 클라이언트 및 사용자로 다운로드 가능한 콘텐츠에 대한 액세스 제한

승인된 애플리케이션 및 클라이언트에 의해 게임 콘텐츠에 대한 액세스를 제한합니다. 다운로드 가능한 게임 콘텐츠를 저장하기 위한 비용 효율적이고 확장 가능한 오리진으로 Amazon S3를 사용하고 플레이어에게 글로벌 성능의 콘텐츠를 제공하기 위해 Amazon CloudFront를 사용하는 것이 좋습니다. 두 서비스 모두 인증된 사용자로 액세스를 제한하는 등 저장된 데이터에 대한 액세스를 제한하기 위한 기본 제공 메커니즘을 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon S3에 저장된 콘텐츠에 대한 액세스 권한 부여

S3에 저장된 콘텐츠에 대한 액세스 권한을 부여해야 하는 경우 고려해야 할 몇 가지 요소가 있습니다. 기본적으로 S3 버킷을 AWS 계정 생성한 만 그 안에 저장된 객체에 액세스할 수 있습니다. 내부 애플리케이션에 대한 액세스 권한을 부여하고 Amazon S3 버킷에 저장된 콘텐츠를 관리하려면 AWS Identity and Access Management (IAM)을 사용하여 적절한 액세스를 제공하는 정책을 생성합니다.

IAM 역할은 Amazon EC2와 같은 서비스에서 호스팅되는 페더레이션 사용자, 시스템 또는 애플리케이션 AWS Lambda과 Amazon EKS 및 Amazon ECS에서 호스팅되는 컨테이너 기반 애플리케이션과 연결할 수 있습니다. 예를 들어 AWS SDK 또는 AWS CLI 를 사용하여 S3 버킷에 게임 콘텐츠 자산을 게시하고 관리할 수 있습니다. 이 사용 사례를 지원하기 위해 S3 버킷에 게임 콘텐츠를 읽고 쓸 수 있는 적절한 액세스 권한이 있는 IAM 역할을 생성하고 소프트웨어 및 스크립트를 호스팅하는 EC2 인스턴스와 연결할 수 있습니다.

버킷 및 특정 객체에 대해 리소스 기반 정책을 정의할 수 있습니다. S3 버킷 정책은 S3 버킷과 연결되며 버킷 및 버킷 내의 객체에 대한 액세스를 제한하고 다른 계정의 Amazon S3 리소스에 대한 액세스 권한을 부여하는 데 사용할 수 있습니다. 예를 들어 여러 팀 또는 별도의 게임 개발 스튜디오가 동일한 게임 콘텐츠에서 작업하고 Amazon S3의 중앙 호스팅 콘텐츠에 대한 동일한 액세스가 필요한 시나리

오에서는 S3 버킷 정책을 사용하여 S3 리소스에 대한 교차 계정 액세스 권한을 정의할 수 있습니다. 각 애플리케이션 또는 애플리케이션 세트에 고유한 이름과 권한이 있는 액세스 포인트를 생성하여 공유 데이터에 대한 데이터 액세스 관리를 간소화할 수 있는 S3 액세스 포인트를 사용하는 것이 좋습니다. Amazon S3 설명서에는 Amazon S3의 액세스 제어를 위한 추가 모범 사례가 포함되어 있습니다.

Granting short-term access to your content

제한된 특정 시간 동안만 액세스해야 하는 경우 콘텐츠에 대한 단기 액세스 권한을 부여하는 임시 URLs을 생성합니다. Amazon S3는 객체 소유자가 버킷 정책을 업데이트하지 않고 Amazon S3의 객체에 대한 시간 제한 액세스 권한을 부여할 수 있도록 미리 서명된 URLs 생성을 지원합니다. 이렇게 하면 액세스 권한이 부여된 최종 사용자 또는 애플리케이션이 계정 또는 IAM 권한을 가질 필요가 없으며 대신 미리 서명된 URL을 사용하여 콘텐츠에 액세스할 수 있습니다.

이는 권한이 있는 플레이어에게 권한이 부여된 다운로드 가능한 콘텐츠에 대한 액세스 권한을 부여하고 제한된 시간 동안의 게임 콘텐츠에 대한 임시 액세스를 제공하는 등 다양한 게임 사용 사례에서 일반적으로 사용되는 모범 사례입니다. 미리 서명된 URLs 사용하여 S3 버킷에 콘텐츠를 업로드할 수 있는 임시 권한을 제공할 수도 있습니다. 예를 들어 미리 서명된 URL을 사용하여 플레이어에게 지원 팀이 플레이어 지원 사례를 해결하는 데 도움이 되도록 클라이언트 로그를 업로드할 수 있는 액세스 권한을 제공하는 것을 고려할 수 있습니다.

콘텐츠 전송 네트워크를 사용하여 콘텐츠에 대한 액세스 제공

애플리케이션, 게임 개발자, 아티스트 및 기타 직원이 개발 및 관리를 위해 S3 버킷의 콘텐츠에 직접 액세스해야 할 수 있지만, 콘텐츠 전송 네트워크를 사용하여 플레이어 또는 다른 사용자가 인터넷을 통해 공개적으로 사용할 수 있는 콘텐츠에 대한 액세스를 제공합니다. 이 접근 방식은 자주 액세스하는 콘텐츠를 캐싱하여 다운로드 성능을 개선하고 비용을 절감합니다. Amazon CloudFront는 콘텐츠를 캐싱하고 플레이어에게 더 가깝게 전달하면서 Amazon S3와 같은 게임의 다운로드 오리진에 대한 부하를 줄여 콘텐츠를 전 세계에 배포할 수 있습니다.

S3 버킷에서 직접 퍼블릭 콘텐츠를 제공하는 대신이 콘텐츠를 비공개로 유지하고 CloudFront를 사용하여 공개적으로 제공하는 것이 좋습니다. CloudFront는 서명된 URLs 또는 서명된 쿠키를 사용하여 플레이어가 프라이빗 콘텐츠(예: 유료 플레이어 전용 새 게임 다운로드)에 액세스하도록 구성할 수 있습니다. 그런 다음 애플리케이션을 개발하여 서명된 URLs을 생성하여 인증된 사용자에게 배포하거나 인증된 사용자에게 서명된 쿠키를 설정하는 쿠키 설정 헤더를 보낼 수 있습니다. 서명된 URLs 또는 서명된 쿠키를 생성하여 파일에 대한 액세스를 제어할 때 URL 및 쿠키가 더 이상 유효하지 않은 종료 날짜 및 시간을 지정할 수 있습니다.

선택적으로 콘텐츠에 액세스하는 데 사용할 수 있는 컴퓨터의 IP 주소 또는 주소 범위를 지정할 수도 있습니다. 이는 특정 게임 개발 스튜디오 파트너 또는 계약업체 네트워크에 대한 액세스를 제한하려는

경우에 유용합니다. 여러 제한된 파일에 대한 액세스를 제공하려는 경우 또는 현재 URLs을 변경하지 않으려는 경우 서명된 쿠키를 사용합니다. 개별 파일에 대한 액세스를 제한하거나 사용자가 쿠키를 지원하지 않는 클라이언트를 사용하는 경우 서명된 URLs을 사용합니다. 서명된 URL은 서명된 쿠키보다 우선합니다.

구현 단계

- IAM 역할 및 버킷 정책을 사용하여 내부 애플리케이션, 팀 또는 교차 계정 시나리오를 위한 S3 버킷에 대한 적절한 액세스 권한을 부여합니다.
- 다운로드 가능한 콘텐츠 또는 클라이언트 로그와 같은 임시 업로드에 적합한 S3 객체에 대한 단기 액세스 권한을 부여하기 위해 미리 서명된 URLs을 생성합니다.
- 서명된 URLs 또는 쿠키와 함께 Amazon CloudFront를 사용하여 인증된 사용자에게 프라이빗 콘텐츠를 더 안전하게 제공

GAMESEC04-BP02 승인된 콘텐츠 전송 네트워크(CDNs.

사용자가 콘텐츠 전송 네트워크를 우회하여 Amazon S3 버킷과 같은 오리진의 콘텐츠에 직접 액세스하지 못하도록 차단합니다. 오리진에 대한 액세스를 승인된 CDNs으로만 제한하는 것이 중요합니다. 이렇게 하면 오리진 외부에서 콘텐츠를 불필요하게 제공하는 데 드는 데이터 전송 비용이 줄어듭니다. 또한 AWS WAF 계층 7 필터링, 보안 관련 HTTP 요청 파라미터의 삽입 및 검사, 분산 서비스 거부(DDoS) 보호와 같은 엣지 보안 제어를 배포할 수 있는 동일한 진입점을 통해 오리진 콘텐츠에 대한 퍼블릭 액세스를 플로우하여 보안 태세를 개선합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon S3 오리진에 대해 이러한 제어를 구현하려면 S3 객체에 대한 요청이 CloudFront 배포에서 시작되었는지 확인하는 Amazon CloudFront 오리진 액세스 ID(OAI)를 사용할 수 있습니다. CloudFront CloudFront 배포 AWS WAF 와 연결하여 계층 7 필터링을 제공합니다. 그러나 추가 CDNs 하나 이상의 사용자 지정 HTTP 헤더를 오리진 요청에 삽입하도록 CDN을 구성할 수 있습니다. 오리진 요청은가 수신 트래픽이 승인된 CDN 공급자에서 시작되었는지 검사 AWS WAF 할 수 있습니다.

이 접근 방식은 오리진이 Application Load Balancer(ALB) 뒤에 호스팅될 때 사용자가 CDN 공급자를 우회하는 것을 방지하는 데에도 유용합니다. ALBs는 계층 7 보호를 위해와 AWS WAF 연결할 수 있습니다. ALB에서 검사할 사용자 지정 HTTP 헤더를 AWS WAF 삽입하여 로드 밸런서로 들어오는 트래픽을 처리하고 검사하도록를 구성할 수 있습니다 AWS WAF.

고객 사례

AnyCompany Games는 오리진 액세스 제한을 구현하여 플레이어가 보안 검사를 우회하거나 적절한 인증 없이 프리미엄 콘텐츠를 획득할 수 있는 무단 직접 액세스로부터 게임 자산, 다운로드 가능한 콘텐츠 및 패치 파일을 보호합니다. 이 접근 방식을 사용하면 중앙 집중식 지점을 통해 콘텐츠 액세스 패턴을 모니터링할 수 있으므로 조정된 공격 또는 무단 콘텐츠 재배포가 있음을 나타낼 수 있는 의심스러운 다운로드 동작을 쉽게 식별할 수 있습니다.

구현 단계

- Amazon CloudFront 오리진 액세스 ID(OAI)를 사용하여 S3 객체에 대한 직접 액세스 제한
- CloudFront 또는 ALB AWS WAF 와 연결하여 계층 7 필터링을 제공하고 DDoS 공격 및 악의적인 요청으로부터 보호합니다.
- Cloudfront에서 사용자 지정 HTTP 헤더를 구성하여 수신 트래픽이 승인된 소스에서 시작되는지 확인합니다.

GAMESEC04-BP03 무단 액세스를 제한하는 지리적 제한 구현

플레이어가 콘텐츠를 요청하면 Amazon CloudFront는 플레이어의 위치에 관계없이 가장 가까운 엣지 로케이션에서 요청된 콘텐츠를 제공합니다. 그러나 전 세계 특정 지역의 사용자가 콘텐츠에 액세스하는 방법을 제한해야 하는 시나리오가 있을 수 있습니다. 예를 들어 country-by-country로 단계별로 콘텐츠를 릴리스하는 롤링 게임 배포 전략이 있거나 국가별 액세스 제어를 준수해야 할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

지리적 차단이라고도 하는 지리적 제한을 사용하여 특정 지리적 위치의 플레이어가 CloudFront 배포를 통해 배포하는 콘텐츠에 액세스하지 못하도록 차단할 수 있습니다. 이 기능을 사용하면 배포와 연결된 파일에 대한 액세스를 제한하고 국가 수준에서 액세스를 제한할 수 있습니다. 또는 타사 지리적 위치 서비스를 사용하여 배포와 연결된 파일의 하위 집합에 대한 액세스를 제한하거나 국가 수준보다 세분화된 수준으로 액세스를 제한할 수 있습니다.

CloudFront 지리적 제한을 사용하면 플레이어가 승인된 국가의 허용 목록에 있는 국가 중 하나에 있는 경우에만 콘텐츠에 액세스하도록 허용할 수 있습니다. 또한 플레이어가 금지된 국가 거부 목록에 있는 국가 중 하나에 있는 경우 플레이어가 콘텐츠에 액세스하지 못하도록 차단할 수 있습니다. 차단된 지리적 위치에서 요청이 수신되면 CloudFront는 403 금지된 HTTP 상태 코드를 플레이어에게 반환합니다. 이는 민감하지 않은 콘텐츠에 적합하며 PII 또는 민감한 게임 아티팩트에 대한 독립 실행형 보호로 사용해서는 안 된다는 점에 유의해야 합니다.

구현 단계

- CloudFront 지리적 제한을 사용하여 국가 수준 허용 또는 거부 목록에 따라 콘텐츠 액세스를 허용하거나 거부합니다.
- 차단된 지리적 위치에서 시작된 요청에 대해 403 금지된 HTTP 상태 코드를 반환합니다.
- 민감한 콘텐츠 또는 PII를 보호하기 위해 지리적 제한에만 의존하지 마세요.

GAMESEC04-BP04 디지털 권한 관리(DRM) 솔루션을 사용하여 콘텐츠에 대한 액세스 제한

디지털 권한 관리(DRM) 솔루션과 같은 강력한 암호화 도구를 사용하여 게임 콘텐츠에 대한 액세스를 제한하는 것이 좋습니다. 이러한 유형의 솔루션을 사용하여 프라이빗 콘텐츠를 암호화하고 복호화 키를 승인된 플레이어에게 배포할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

플레이어가 게임 콘텐츠를 조기에 다운로드하도록 허용하고 싶지만 미리 정해진 시간까지 콘텐츠에 액세스하거나 재생할 수 없도록 하려는 상황에서는 DRM 솔루션을 사용하는 것이 좋습니다. 예를 들어 플레이어가 게임을 사전 주문하고 암호화된 파일 다운로드를 조기에 자동으로 시작하도록 게임 클라이언트를 구성할 수 있는 상황에서는 일반적입니다. 이 전략은 게임이 공식적으로 릴리스되면 게임이 다운로드되고 재생 준비가 되었는지 확인합니다. 게임이 릴리스되면 플레이어의 게임 클라이언트는 이전에 다운로드한 파일을 해독하고 게임 플레이를 시작할 수 있도록 DRM 백엔드 솔루션에서 해독 키를 요청할 수 있습니다.

DRM 시스템은 승인된 플레이어가 게임을 다운로드하고 설치한 후 게임의 무단 재배포 및 조작을 차단하는 데에도 사용됩니다. DRM 시스템은 암호화 키를 교환하고 플레이어가 복호화 키를 검색할 수 있도록 권한을 부여하기 위해 오리진과 통합해야 합니다. 상용 DRM 공급자는 다양한 디바이스에 대한 기능과 지원을 갖춘 다양한 솔루션을 제공합니다.

구현 단계

- DRM 솔루션을 사용하여 프라이빗 게임 콘텐츠를 암호화하고 복호화 키를 승인된 플레이어에게 배포합니다.
- 미리 정렬된 게임에 대해 암호화된 파일의 사전 다운로드를 활성화하여 릴리스 시 복호화 키를 사용하여 액세스를 잠금 해제합니다.

- DRM 시스템을 오리진과 통합하여 암호화 키를 관리하고 콘텐츠의 무단 재배포 또는 조작을 차단합니다.

탐지

GAMESEC05: 게임 내에서 플레이어 사용 동작을 어떻게 모니터링하고 분석하나요?

플레이어 사용 동작을 모니터링하고 분석하는 것은 게임 스튜디오에서 필수적입니다. 게임 무결성과 플레이어 안전을 손상시킬 수 있는 보안 위협, 부정 행위 및 기타 형태의 모욕적인 행동을 탐지할 수 있기 때문입니다. 비정상적인 진행률, 비정상적인 게임 내 트랜잭션 또는 의심스러운 커뮤니케이션 동작과 같은 패턴을 추적하면 플레이어 경험에 상당한 영향을 미치기 전에 잠재적 치터, 사기 계정 또는 조정된 위협을 식별할 수 있습니다.

모범 사례

- GAMESEC05-BP01 플레이어 동작을 모니터링하기 위한 포괄적인 데이터 수집 전략 구현
- GAMESEC05-BP02 플레이어 사용 로그를 수집, 저장 및 분석하여 부적절한 동작 감지

GAMESEC05-BP01 플레이어 동작을 모니터링하기 위한 포괄적인 데이터 수집 전략 구현

긍정적인 플레이어 경험을 유지하려면 포괄적인 데이터 수집 및 분석 전략을 구현합니다. 관련 데이터를 캡처, 저장 및 분석하면 플레이어가 게임의 기능 및 서로 상호 작용하는 방식에 대한 인사이트를 얻을 수 있습니다. 이 데이터 기반 접근 방식은 의사 결정을 안내하고, 플레이어 참여 및 유지를 개선하고, 수익화 전략을 최적화하고, 궁극적으로 전반적인 플레이어 경험을 개선할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

데이터 수집 시스템을 구현하여 게임 플레이 세션, 진행 상황, 업적, 구매, 게임 요소와의 상호 작용, 소셜 활동과 같은 관련 플레이어 작업을 캡처하고 로깅합니다. 서버 로드, 네트워크 트래픽 및 오류 로그와 같은 서버 측 데이터를 수집하여 기술 성능을 모니터링하고 잠재적 문제를 식별합니다. 설문 조사, 포럼, 지원 티켓 및 소셜 미디어 채널을 통해 플레이어 피드백을 수집하여 이들의 경험과 선호도를 파악합니다.

게임 데이터를 저장할 때 중앙 집중식 데이터 웨어하우스 또는 데이터 레이크를 설정하여 수집된 데이터를 저장 및 구성하고 데이터 정리, 변환 및 집계를 위한 파이프라인을 구현하여 효율적인 분석을 위해 데이터를 준비합니다.

데이터를 저장한 후 이를 분석하여 데이터 시각화 도구를 통해 플레이어 보존 및 이탈, 수익화 전략, 기능 사용과 같은 인사이트를 얻습니다.

구현 단계

- 플레이어 작업, 서버 측 지표 및 피드백을 캡처하고 기록하여 상호 작용 및 기술 성능을 모니터링합니다.
- Amazon Redshift 또는 S3 데이터 레이크와 같은 중앙 집중식 데이터 웨어하우스를 사용하여 분석을 위해 게임 데이터를 저장, 정리, 변환 및 구성합니다.
- Amazon Quicksight와 같은 시각화 도구를 사용하여 수집된 데이터를 분석하여 플레이어 보존, 수익화 및 기능 사용에 대한 인사이트를 얻습니다.

GAMESEC05-BP02 플레이어 사용 로그를 수집, 저장 및 분석하여 부적절한 동작 감지

플레이어가 게임의 기능을 사용하는 방식과 플레이어가 다른 플레이어와 상호 작용하는 방식을 이해하기 위해 게임을 계측하여 로그를 수집합니다. 그런 다음 플레이어 경험을 저하시킬 수 있는 무단 활동을 차단할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

Amazon Amazon CloudWatch OpenSearch Service와 같은 로깅 솔루션을 사용하거나 Datadog, Sumo Logic, New Relic, Honeycomb.io 또는 Splunk와 같은 파트너의 솔루션을 통해 Game Analytics 파이프라인으로 AWS 구조화된 로그 이벤트를 전송합니다. 플레이어의 특정 작업을 조사해야 하는 시기를 감지하는 데 사용할 수 있도록 이러한 플레이어 사용 로그를 구성합니다.

데이터를 캡처한 후에는 부적절한 사용 동작을 감지하는 도구를 구현하는 것이 좋습니다. 예를 들어 게임에 게임 내 플레이어 메시징, 음성 채팅 또는 온라인 포럼과 같은 소셜 기능이 있는 경우 조정 목적으로 분석할 수 있는 형식으로 이러한 플레이어 참여의 로그를 저장합니다.

레코딩을 Amazon S3로 내보내고 Amazon Transcribe를 사용하여 오디오 스피치를 처리를 위해 저장할 수 있는 텍스트 형식으로 변환하도록 게임의 음성 채팅 기능을 구성합니다. 또는 게임 백엔드 음

성 채팅 서비스를 Transcribe API와 직접 통합하여 스트리밍 오디오를 실시간으로 트랜스크립션하여 실시간 스트리밍 트랜스크립션을 수행할 수 있습니다. 조절 팀은 콘텐츠를 수동으로 검토할 수 있으며, 콘텐츠가 표준 형식이면 AWS AI/ML 서비스를 사용하여 조절을 자동으로 수행할 수도 있습니다. Amazon Comprehend를 사용하여 자연어 처리(NLP)를 수행하여 비정형 텍스트에서 정보를 발견할 수 있습니다. 이를 통해 대화를 관련 주제로 분류 및 구성하고 비속어와 같은 부적절한 행동을 식별할 수 있습니다.

구현 단계

- 플레이어 사용 로그를 수집, 저장 및 분석합니다.
- 인공 지능 및 기계 학습을 위한 AWS 서비스를 사용하여 플레이어 사용 로그를 보다 효율적으로 검토하고 인사이트를 얻을 수 있습니다.

인프라 보호

게임 워크로드에 적용되는 보안을 위한 인프라 보호의 모범 사례는 Well-Architected Framework 백서를 참조하세요.

GAMESEC06: 인프라 위협을 어떻게 모니터링하고 대응하나요?

인프라 위협 모니터링 및 대응은 게임 스튜디오에 매우 중요합니다. 인프라는 수백만 명의 동시 플레이어를 지원하고, 실제 화폐 트랜잭션을 처리하고, 귀중한 플레이어 데이터와 독점 게임 콘텐츠를 저장하는 백본을 나타내기 때문입니다. 게임 스튜디오는 플레이어 경험과 비즈니스 운영의 무결성을 모두 유지할 수 있는 모니터링 시스템 및 인시던트 대응 프로세스를 구현해야 합니다.

모범 사례

- GAMESEC06-BP01 인프라에 대한 위협을 탐지하고 대응하기 위한 도구 사용
- GAMESEC06-BP02 인공 지능 및 기계 학습 도구를 사용하여 인프라 보호 전략의 측면 자동화
- GAMESEC06-BP03 시스템 수준 로그의 인사이트를 사용하여 인프라 보호 전략을 지속적으로 개선합니다.

GAMESEC06-BP01 인프라에 대한 위협을 탐지하고 대응하기 위한 도구 사용

AWS 환경 내에서 악의적인 활동 및 무단 동작을 지속적으로 모니터링하려면 Amazon GuardDuty를 사용하는 것이 좋습니다. GuardDuty는 환경 내에서 계정 동작, 네트워크 활동 및 데이터 액세스 패턴을 모니터링하여 위협을 식별합니다. CloudTrail 이벤트 로그, Amazon VPC 흐름 로그 및 DNS 로그와 같은 여러 데이터 소스의 이벤트를 분석하여 잠재적 위협이 있는지 확인합니다. Amazon CloudWatch Events 및 Lambda와 통합하면 추가 분석을 위해 GuardDuty 알림을 관련 보안 팀에 자동으로 전달할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS Security Hub CSPM는의 보안 상태를 포괄적으로 파악하고 보안 업계 표준 및 모범 사례를 기준으로 환경을 AWS 확인합니다. Security Hub CSPM은 여러 AWS 계정서비스 및 지원되는 타사 파트너 제품에서 보안 데이터를 수집하고 보안 추세를 분석하고 우선 순위가 가장 높은 보안 문제를 식별합니다. Amazon GuardDuty와 Security Hub CSPM의 통합을 통해 GuardDuty에서 Security Hub CSPM으로 조사 결과를 전송할 수 있습니다. 그런 다음 Security Hub CSPM은 이러한 결과를 보안 태세 분석에 포함할 수 있습니다.

악의적인 행위자가 계정을 인계하고 게임을 치트하기 위해 봇을 사용하는 것이 일반적입니다. WAF Bot Control을 사용하면 과도한 리소스를 소비하거나, 지표를 왜곡하거나, 가동 중지를 일으키거나, 기타 원치 않는 활동을 수행할 수 있는 일반적이고 널리 사용되는 봇 트래픽을 파악하고 제어할 수 있습니다.

랜섬웨어는 시스템 및 데이터 세트에 대한 무단 액세스를 얻고 해당 데이터를 암호화하여 합법적인 플레이어의 액세스를 차단하도록 설계된 악성 코드입니다. 랜섬웨어가 플레이어를 시스템에서 잠그고 민감한 데이터를 암호화한 후 사이버 범죄자는 데이터 잠금을 해제하기 위해 복호화 키를 제공하기 전에 랜섬웨어를 요구합니다. 악의적인 이벤트로 인해 조직이 완전히 종료되어 상당한 비용이 발생하고 비즈니스 생산성이 저하될 수 있습니다. 인시던트 발생 전, 발생 중, 발생 후 랜섬웨어와 싸우는 능력을 강화하기 위해 적용할 수 있는 모범 사례는 랜섬웨어로부터 AWS 클라우드 환경 보호를 참조하세요.

게임은 Amazon Amazon Connect Amazon Lex를 사용하는 채팅 봇과 같은 콜센터를 통해 플레이어 지원 에이전트에게 연락할 수 있는 기능을 플레이어에게 제공할 수 있습니다. Amazon Connect는 실시간 및 녹음된 대화를 모니터링할 수 있는 지원을 제공합니다. https://docs.aws.amazon.com/connect/latest/adminguide/monitoring-amazon-connect.html Amazon Lex로 구축된 플레이어와 플레이어 지원 채팅 봇 간의 상호 작용을 분석하려면 앞서 설명한 대로 Amazon S3로 내보내고 분석할 수 있는 Amazon CloudWatch Logs에 이러한 상호 작용의 대화 로그를 저장할 수 있습니다.

마지막으로 인프라 보호 전략의 일환으로 침투 테스트 연습을 수행합니다. 이러한 평가를 사내에서 수행하든 AWS 파트너를 통해 수행하든 상관없이 AWS 침투 테스트를 위한 고객 지원 정책을 준수하세요.

구현 단계

- Amazon GuardDuty를 사용하여 위협에 대한 계정 동작, 네트워크 활동 및 데이터 액세스 패턴을 모니터링하고 Security Hub CSPM과 통합하여 통합 보안 보기를 제공합니다.
- AWS WAF Bot Control을 구현하여 리소스와 플레이어 경험에 해를 끼칠 수 있는 봇 트래픽을 감지하고 완화할 수 있습니다.
- AWS 고객 지원 정책에 따라 정기적으로 침투 테스트 연습을 수행하여 보안 태세를 평가하고 강화합니다.

GAMESEC06-BP02 인공 지능 및 기계 학습 도구를 사용하여 인프라 보호 전략의 측면 자동화

Amazon Lookout for Metrics는 기계 학습을 사용하여 비즈니스 및 운영 데이터의 이상을 자동으로 감지 및 진단하고 비즈니스에 가장 중요한 지표를 더 빠르고 정확하게 모니터링합니다. 또한이 서비스를 사용하면 수익, 로그인, 트랜잭션 또는 보존의 갑작스러운 감소와 같은 이상 현상의 근본 원인을 쉽게 진단할 수 있습니다. 게임 개발자는 ML 경험을 통해를 설정할 필요가 없으며 Amazon S3, Amazon CloudWatch, Amazon RDS, Amazon Redshift 및 많은 SaaS 애플리케이션을 비롯한 널리 사용되는 데이터 소스에 연결할 수 있습니다. 예를 들어 Amazon Lookout for Metrics를 Game Analytics Pipeline 및 기타 데이터 소스와 통합하여 이상 탐지를 위한 동작 분석을 시작할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

또는 Amazon SageMaker AI AI를 사용하여 사용자 지정 기계 학습 모델을 빌드, 훈련 및 호스팅하여 콘텐츠 조절, 유해성 탐지, 치트 탐지, 사기 탐지 등과 같은 사용 사례를 해결할 수 있습니다.

고객 사례

AnyCompany Games는 Amazon Lookout for Metrics를 사용하여 악의적인 행위자의 위협을 나타낼 수 있는 서버 성능, 플레이어 로그인 시도 또는 트랜잭션 볼륨의 비정상적인 패턴을 자동으로 감지합니다. 또한 Amazon SageMaker AI를 사용하여 네트워크 트래픽 패턴과 플레이어 동작을 지속적으로 분석하여 가상 경제를 악용하려는 봇 네트워크와 같은 조정된 위협을 식별하는 사용자 지정 기계 학습 모델을 개발했습니다.

이 자동화된 접근 방식을 통해 보안 팀은 수천 개의 지표를 수동으로 모니터링하는 대신 실제 위협을 조사하고 대응하는 데 집중할 수 있으며, 새로운 위협 패턴이 게임 가용성 또는 플레이어 안전성에 상당한 영향을 미치기 전에 탐지되고 해결되도록 할 수 있습니다.

구현 단계

- Amazon Lookout for Metrics를 사용하여 주요 비즈니스 및 운영 데이터의 이상을 자동으로 감지하고 진단할 수 있습니다.
- Amazon Lookout for Metrics를 Game Analytics Pipeline, Amazon S3 또는 CloudWatch와 같은 데이터 소스와 통합하여 수익, 로그인 및 보존과 같은 지표를 모니터링합니다.
- Amazon SageMaker AI를 사용하여 치트 탐지, 사기 방지 및 콘텐츠 조절과 같은 고급 사용 사례를 위한 사용자 지정 기계 학습 모델을 구축, 훈련 및 호스팅할 수 있습니다.

GAMESEC06-BP03 시스템 수준 로그의 인사이트를 사용하여 인프라 보호 전략을 지속적으로 개선합니다.

S3 서버 액세스 로그, CloudFront 액세스 로그 및 ALB 액세스 로그와 같은 관련 서비스에서 시스템 수준 로그를 캡처하고 저장합니다. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html 이러한 로그는 계정의 S3 버킷에 저장할 수 있으며, 게임 백엔드 내에서 구성했을 수 있는 IP 주소, 요청 헤더, 관련 요청 조작 및 필터링과 같은 연결 세부 정보를 포함한 시스템 수준 정보와 게임 내에서 플레이어 사용 정보를 연결하는 데 유용합니다. 이러한 로그를 앞서 언급한 것과 동일한 로깅 솔루션으로 전송할 수 있으며, 로그를 Amazon S3 외부로 이동할 필요 없이 Amazon Athena에서 SQL 쿼리를 사용하여 분석할 수 있습니다. Amazon S3

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

S3용 Access Analyzer는 버킷 액세스 정책을 모니터링하는 기능으로, 정책이 Amazon S3 리소스에 대한 의도된 액세스만 제공하도록 합니다. S3용 Access Analyzer는 버킷 액세스 정책을 평가하고 잠재적으로 의도하지 않은 액세스가 있는 버킷을 검색하고 신속하게 수정할 수 있습니다.

구현 단계

- 위협 탐지 및 인시던트 대응을 위한 AWS 서비스를 사용하여 인프라 보호 전략의 측면을 자동화합니다.
- 인공 지능 및 기계 학습을 위한 시스템 수준 로그 및 AWS 서비스를 통해 인프라 보호에 대한 인사이트를 얻습니다.

데이터 보호

게임을 개발하고 설계할 때는 스튜디오에서 수집하는 데이터 유형과 이를 보호하기로 결정한 방법을 고려하세요. 이 보안 측면 내에서 살펴볼 주제는 다음과 같습니다.

- 데이터를 식별하고 분류하도록 선택한 방법
- 저장 데이터를 보호하는 방법
- 전송 중 데이터를 보호하는 방법

Games Lens와 관련된 데이터 보호 모범 사례는 없습니다. 보안을 위한 데이터 보호 모범 사례는 Well-Architected Framework 백서를 참조하세요.

인시던트 대응

GAMESEC07: 플레이어의 불법 행위 및 모욕적인 행동에 대응하기 위한 정책을 어떻게 정의하고 적용하고 있나요?

플레이어 불법 행위와 모욕적인 행동은 플레이어의 경험에 상당한 영향을 미칠 수 있습니다. 플레이어 동작이 잘못되면 합법적인 플레이어가 이탈하여 플레이어 유지율 감소, 게임 내 구매로 인한 수익 감소, 게임의 평판과 향후 매출을 손상시킬 수 있는 부정적인 리뷰가 발생할 수 있습니다.

플레이어 간에 긍정적인 행동을 장려하는 정책을 정의하고 이러한 정책을 적용하는 방법을 결정합니다.

모범 사례

- GAMESEC07-BP01 악의적인 행위자와 모욕적인 행동을 처리하기 위한 인시던트 대응 계획 구현
- 악의적인 행위자와 연결된 GAMESEC07-BP02 금지 계정

GAMESEC07-BP01 악의적인 행위자와 모욕적인 행동을 처리하기 위한 인시던트 대응 계획 구현

게임에서 악의적인 행위자와 모욕적인 행동에 대응하기 위한 행동 계획을 수립합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

플레이어를 일시 중지하거나 영구적으로 금지해야 하는 시기 및 일시 중지된 플레이어의 자격 증명을 비활성화하는 기간과 같은 요소를 고려합니다.

고객 사례

AnyCompany Games는 부적절한 채팅 메시지와 같은 사소한 위반으로 인해 24시간 계정 자동 일시 중지가 발생하는 계층형 인시던트 대응 시스템을 생성하는 반면, 사기 또는 괴롭힘과 같은 보다 심각한 위반으로 인해 진행자의 필수 검토와 함께 7일 즉시 일시 중지가 트리거됩니다.

또한 AnyCompany Games는 반복 위반자가 점진적으로 더 긴 일시 중지에 직면하는 에스컬레이션 절차를 수립합니다. 잘못 플래그가 지정된 플레이어가 자격 증명 확인 요구 사항을 통해 보안을 유지하면서 자동화된 작업에 이의를 제기하도록 허용하는 어필 프로세스를 생성합니다.

악의적인 행위자와 연결된 GAMESEC07-BP02 금지 계정

게임에서 모욕적인 행동을 완화하지 않으면 다른 사람의 게임 경험에 부정적인 영향을 미칠 수 있으므로 최대한 빨리 완화해야 합니다. 서비스 약관을 위반한 것으로 확인된 악의적인 행위자에게 금지 또는 기타 형태의 제한을 부과하는 프로세스를 구현합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

일반적으로 이러한 유형의 제한을 적용하기 위한 상황을 결정하기 위한 규칙 및 평가 프로세스는 플레이어 커뮤니티 팀 또는 조직 내 신뢰 및 안전 팀과 같은 직원이 결정합니다. 악의적인 액터에 플래그를 지정한 후 미리 결정된 워크플로를 실행하여 식별된 플레이어에 대해 조치를 취합니다.

예를 들어 AWS Step Functions 및 AWS Lambda 함수를 사용하여 플레이어 계정 배치를 입력으로 수락하는 자동화된 워크플로를 실행할 수 있습니다. 그런 다음 워크플로는 플레이어 계정, 금지 이유 및 기간에 대한 세부 정보를 포함할 수 있는 Bans라는 Amazon DynamoDB 테이블의 항목을 업데이트합니다.

게임 및 계정 관리 시스템의 설계와 악의적인 행위자로부터 발생하는 침해 유형에 따라 계정 관리 시스템과 별도의 레코드 차단 시스템을 유지 관리합니다. 계정 관리 시스템에서 플레이어의 계정을 끄지 않고 게임 플레이 기능을 끄도록 선택할 수 있습니다. 이는 플레이어의 계정 자격 증명을 사용하여 다양한 서비스 약관 또는 정책으로 여러 게임에 액세스하는 경우에 유용할 수 있습니다.

구현 단계

- 악의적인 행위자의 모욕적인 행동에 대응하기 위한 정책을 정의하고 적용합니다.

- AWS 서비스를 사용하여 악의적인 액터에 대한 응답을 자동화합니다.

리소스

- AWS 보안 인시던트 대응 기술 가이드
- AWS Machine Learning 블로그: Amazon Rekognition Face Liveness를 사용하여 실제 및 실제 사용자를 감지하고 악의적인 행위자를 방지
- AWS 게임 솔루션: 커뮤니티 상태

애플리케이션 보안

GAMESEC08: CI/CD 파이프라인을 어떻게 보호하나요?

게임 개발 CI/CD 파이프라인은 일반적으로 가용성이 높은 소스 제어 서버 및 스토리지, 빌드를 실행하는 컴퓨팅 리소스, 자동 테스트를 수행하는 소프트웨어, 개발 머신의 적절한 네트워크 연결로 구성됩니다. 민감한 정보를 보호하고, 코드 무결성을 유지하고, 신뢰할 수 있는 릴리스를 유지하려면 CI/CD 파이프라인을 보호하는 것이 중요합니다. 거버넌스 및 가드레일을 임베딩하면 개발자 민첩성을 높이는 동시에 모범 보안 사례를 유지할 수 있습니다.

게임은 결제 처리를 처리하고, 개인 정보를 저장하고, 실제 가치가 있는 가상 경제를 유지하는 경우가 많기 때문에 개발 프로세스의 보안 위반으로 인해 상당한 재정적 손실, 규제 처벌 및 플레이어 신뢰 상실이 발생할 수 있습니다.

조직은 보호 장치를 통합하여 소프트웨어 제공 프로세스에 대한 가시성과 제어를 유지하여 신속한 인시던트 대응을 지원하고 안전한 코딩 관행의 문화를 조성합니다.

모범 사례

- GAMESEC08-BP01 CI/CD 파이프라인의 모든 단계에서 보안 적용

GAMESEC08-BP01 CI/CD 파이프라인의 모든 단계에서 보안 적용

액세스 제어, 업무 분리 및 감사 추적과 같은 가드레일은 무단 액세스 또는 악의적인 활동에 대한 보호를 제공합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

사람, 프로세스 및 기술도 파이프라인을 보호해야 합니다. 코드에 가장 가까운 사람은 보안 코딩 관행을 수립하고 이를 따라야 합니다. 프로세스를 지속적으로 반복하여 파이프라인 전반의 보안 수준에 일관성이 있는지 확인합니다. 마지막으로 기술을 구현하여 모범 사례와 프로세스가 우회되지 않는지 확인합니다.

고객 사례

AnyCompany Games는 플레이어 결제 데이터를 처리하는 구성 요소에 대해 보안 팀원의 필수 코드 검토를 요구하면서 고위 개발자만 치트 방지 시스템 코드의 변경을 승인할 수 있는 역할 기반 액세스 제어를 구현합니다.

CI/CD 파이프라인은 위협 모델 검증 검사를 자동으로 실행하여 플레이어 거래 마켓플레이스와 같은 새로운 기능이 항목 중복 악용 또는 사기 거래 시도와 같이 이전에 식별된 공격 벡터에 대해 테스트되는지 확인합니다.

구현 단계

- 최소 권한 원칙을 기반으로 사용자에게 권한을 제공합니다.
- AWS CloudTrail 를 사용하여 파이프라인에 사용되는 서비스에서 이루어진 API 직접 호출을 감사합니다.
- 커밋 전 후크를 사용하여 코드가 일반 관행 및 회사 정책을 따르고 있는지 확인합니다.

보안 자동화

GAMESEC09: CI/CD 파이프라인 내에서 보안을 자동화하려면 어떻게 해야 하나요?

CI/CD 파이프라인 내에 보안 조치를 통합하여 개발 수명 주기 전반에 걸쳐 강력한 보안 태세를 유지합니다. 이 프로세스는 파이프라인의 보안을 유지하는 것과 동일한 많은 이점을 제공합니다. 보안 CI/CD 파이프라인을 사용하면 보안 이벤트로 인해 게임 개발 타임라인이 지연될 가능성이 줄어듭니다.

CI/CD 파이프라인에 보안을 제공하려면 개발 주기의 모든 단계에서 보안 모범 사례와 도구를 구현해야 합니다. 파이프라인에 보안이 있으면 보안 검토 시간도 줄일 수 있습니다.

CI/CD 파이프라인 내에서 보안을 자동화하는 것은 코드가 변경될 때마다 보안 제어가 일관되게 구현되고 테스트되는지 확인하는 데 특히 중요합니다. 적절한 도구와 자동화를 구현하면 안전하고 안전한 게임을 제공할 수 있습니다.

모범 사례

- GAMESEC09-BP01 도구 및 자동화를 통합하여 평균 보안 검토 시간 단축

GAMESEC09-BP01 도구 및 자동화를 통합하여 평균 보안 검토 시간 단축

조직은 보안 취약성을 식별하기 위해 정적 애플리케이션 보안 테스트(SAST) 및 동적 애플리케이션 보안 테스트(DAST)와 같은 다양한 도구와 서비스를 사용할 수 있습니다. SAST는 소스 코드를 검토하고 보안 취약성을 확인하는 방법입니다. DAST는 코드를 테스트하는 블랙박스 방법으로, 소스 코드를 확인하지 않고 애플리케이션을 테스트합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

조직에서 사용할 수 있는 또 다른 도구는 타사 또는 오픈 소스 종속성의 보안을 평가하는 소프트웨어 구성 분석(SCA)입니다. 보다 수동적인 접근 방식을 위해 파이프라인 전체에서 보안 코드 검토를 구현할 수 있습니다.

고객 사례

AnyCompany Games는 SAST 도구를 사용하여 개발 프로세스 중에 잠재적 보안 결함에 자동으로 플래그를 지정합니다. 또한 DAST 도구를 사용하여 실행 중인 게임 빌드에 대한 위협을 시뮬레이션하여 보안 제어가 의도한 대로 작동하는지 검증합니다. 또한 AnyCompany Games는 종속성 스캔 도구를 개발 프로세스에 통합하여 타사 라이브러리 및 게임 엔진에서 알려진 취약성을 자동으로 식별합니다.

구현 단계

- Amazon CodeGuru를 SAST 도구로 사용합니다.
- OWASP Dependency Check, SonarQube 또는 OWASPZap과 같은 오픈 소스 도구를 사용합니다.

리소스

- 개발자를 위한 보안

위협 모델링

GAMESEC10: 위협 모델링을 조직의 애플리케이션 개발 수명 주기에 어떻게 통합하나요?

위협 모델링은 애플리케이션에 대한 잠재적 위협을 식별하고 우선순위를 지정하며 이를 완화하는 데 사용할 수 있는 솔루션을 결정하는 프로세스입니다. 게임이 민감한 사용자 데이터와 실제 화폐 트랜잭션을 처리하는 복잡하고 연결된 시스템으로 발전함에 따라 이러한 관행이 점점 더 중요해지고 있습니다.

위협 모델링을 초기 설계 단계뿐만 아니라 게임이 계속 성장하고 발전함에 따라 게임의 보안을 지원하기 위한 지속적인 연습으로 통합합니다.

모범 사례

- GAMESEC10-BP01 애플리케이션 개발 수명 주기 전반에 걸쳐 위협 모델링 연습을 완료하는 시기와 방법 결정

GAMESEC10-BP01 애플리케이션 개발 수명 주기 전반에 걸쳐 위협 모델링 연습을 완료하는 시기와 방법 결정

위협 모델링에 접근하는 가장 좋은 방법은 없습니다. 이 작업을 수행하는 시기와 방법에 대한 세부 정보는 게임 스튜디오의 고유한 요구 사항에 따라 달라집니다. 예를 들어 스튜디오의 크기에 따라 위협 모델링 프로세스의 하나 또는 여러 측면에 관여하는 팀원이 있을 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

AWS 보안 블로그는 다음과 같이 위협 모델링 전략을 고안할 때 염두에 두어야 할 고려 사항에 대한 개요를 제공합니다.

- 위협 모델링에 관여해야 하는 팀원과 페르소나
- 사용할 적절한 워크플로 도구를 결정하는 방법
- 위협 모델링의 다양한 측면에 대한 소유권을 결정하는 방법
- 워크로드 설계에서 사용할 보안 제어를 식별하고 평가하는 방법

고객 사례

AnyCompany Games는 플레이어 데이터, 게임 코드 및 알고리즘, 게임 내 통화, 사용자 생성 콘텐츠, 릴리스되지 않은 콘텐츠 또는 독점 엔진과 같은 지적 재산과 같은 중요한 자산을 카탈로그화하는 것부터 시작합니다. 이들은 불공정한 이점을 원하는 치터, 개인 또는 금융 데이터를 도용하려는 악의적인 행위자, 게임 플레이를 방해하려는 악의적인 사용자와 같은 다양한 유형의 잠재적 악의적인 행위자를 고려합니다.

개발 프로세스 전반에 걸쳐 AnyCompany Games는 위협 모델을 사용하여 보안 코딩 관행을 안내하고 테스트 전략에 영향을 주어 고위험 영역에 집중합니다. 게임이 출시되기 전에 포괄적인 위협 모델링 검토를 수행하여 예상 플레이어 로드 및 무단 액세스 시도에 대한 준비 상태를 평가하고 인시던트 대응 절차를 준비합니다.

구현 단계

- CI/CD 파이프라인의 모든 단계에서 가드레일을 구현합니다.
- 자동화 및 도구를 사용하여 애플리케이션 보안 검토의 효율성을 개선합니다.
- 위협 모델링을 애플리케이션의 보안을 개선하기 위한 프로세스로 사용합니다.

리소스

- AWS 보안 블로그: 위협 모델링에 접근하는 방법
- NIST: 데이터 중심 시스템 위협 모델링 가이드
- 빌더를 위한 올바른 위협 모델링 - AWS Skill Builder 가상 자기 주도 훈련
- 빌더를 위한 위협 모델링 - AWS 워크숍

리소스

보안과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

관련 문서:

- 일반적인 Amazon Cognito 시나리오
- 서명된 URLs 사용
- 채널 흐름을 사용하여 Amazon Chime SDK 메시징의 메시지에서 욕설 및 민감한 콘텐츠 제거
- Amazon GameLift의 보안

- Amazon CloudFront를 사용한 보안 콘텐츠 전송
- 보안 대응 가이드
- AWS DDoS 복원력 모범 사례
- 랜섬웨어로부터 AWS 클라우드 환경 보호

관련 파트너 솔루션:

- Datadog
- Sumo Logic
- Splunk
- Honeycomb.io
- New Relic
- AWS Marketplace - DRM 솔루션

관련 교육 자료:

- Amazon Cognito 시작하기
- 보안 자기 주도 훈련

신뢰성

신뢰성 원칙에는 인프라 또는 서비스 중단으로부터 복구하고, 수요를 충족하기 위해 컴퓨팅 리소스를 동적으로 획득하고, 잘못된 구성 또는 일시적인 네트워크 문제와 같은 중단을 완화하는 시스템의 기능이 포함됩니다.

중점 영역

- 설계 원칙
- 기본
- 워크로드 아키텍처
- 변경 관리
- 장애 관리
- 리소스

설계 원칙

AWS Well-Architected Framework 백서의 설계 원칙 외에도 게임 워크로드에 대한 클라우드의 신뢰성을 높일 수 있는 설계 원칙은 다음과 같습니다.

- 비즈니스 프로젝션을 충족하는 데 필요한 피크 플레이어 동시성 및 시스템 확장성 목표의 기준을 설정합니다. 게임을 시작하기 전과 라이브 게임 운영 중에 이러한 프로젝션을 충족하는 시스템 확장성을 위한 목표 목표를 설정하기 위해 피크에 도달할 것으로 예상되는 동시 플레이어 수에 대한 추정치를 개발합니다. 이렇게 하면 게임의 신뢰성을 위한 기준을 만드는 데 도움이 됩니다. 조정 시스템이 활성 플레이어 세션을 정상적으로 관리하는지 확인하여 가용성에 영향을 주지 않고 수요 변화를 자동으로 수용하도록 조정 정책을 정의합니다.
- 신뢰성과 플레이어 경험에 미치는 영향 측정: 게임 상태를 나타내는 핵심 성과 지표(KPIs)를 정의합니다. 인프라 및 게임 기능의 변경 사항이 신뢰성에 미치는 영향을 모니터링합니다.

기본

신뢰성을 달성하려면 시스템에 수요 또는 요구 사항의 변화를 처리하기 위한 메커니즘과 함께 잘 계획된 기반과 모니터링이 있어야 합니다. 시스템은 장애를 감지하고 자동으로 복구하도록 설계되어야 합니다.

Games Lens와 관련된 파운데이션 모범 사례는 없습니다. 게임 워크로드에 적용되는 신뢰성을 위한 기초의 모범 사례는 Well-Architected Framework 백서를 참조하세요.

워크로드 아키텍처

GAMEREL01: 게임 아키텍처가 클라우드의 복원력을 활용하고 있습니까?

AWS 인프라는 리전 및 가용 영역을 중심으로 구축됩니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공하며,이 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워킹을 사용하여 연결됩니다. 이러한 구조를 사용하여 신뢰성 목표를 중심으로 워크로드를 설계할 수 있습니다.

모범 사례

- GAMEREL01-BP01 여러 가용 영역 및 리전에 게임 인프라를 분산하여 복원력 향상

GAMEREL01-BP01 여러 가용 영역 및 리전에 게임 인프라를 분산하여 복원력 향상

현지화된 인프라 장애가 플레이어에게 미치는 영향을 최소화하려면 예기치 않은 장애를 견딜 수 있을 만큼 충분한 독립 위치에 인프라 배포를 균일하게 배포하는 동시에 플레이어 요구 사항을 충족할 수 있는 충분한 용량을 확보해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임 인프라를 배포할 때는 플레이어 경험을 방해하지 않고 하나 이상의 가용 영역에 대한 중단을 견딜 수 있도록 리전의 여러 가용 영역에 용량을 균일하게 분산하는 것이 좋습니다. 웹 애플리케이션과 같은 게임 백엔드 서비스는 여러 가용 영역에서 로드 밸런싱하거나 설계상 리전 고가용성을 제공하는 AWS Lambda 및 Amazon API Gateway와 같은 관리형 서비스를 사용하여 구축해야 합니다. 마찬가지로 캐시, 데이터베이스, 메시지 대기열, 스토리지 솔루션과 같은 상태를 유지하는 구성 요소는 Amazon S3, DynamoDB, Amazon SQS와 같은 서비스에서 설계로 제공되며 다른 서비스에서 구성할 수 있는 여러 가용 영역에서 데이터의 지속적인 지속성을 제공하도록 설계해야 합니다. DynamoDB

복원력을 위해 게임 서버 호스팅 아키텍처를 설계할 때 내 가용 영역에 게임 서버 플릿을 균일하게 배포 AWS 리전 하여 리전에서 사용 가능한 컴퓨팅 용량에 대한 액세스를 극대화하고 가용 영역 장애의

영향 범위를 줄입니다. 예를 들어 가용 영역을 사용하도록 Amazon EC2 Auto Scaling을 구성할 수 있습니다. EC2 인스턴스가 비정상이 되면 EC2 Auto Scaling은 인스턴스를 교체하고 하나 이상의 가용 영역을 사용할 수 없게 되면 다른 가용 영역으로 인스턴스를 시작할 수 있습니다.

인증과 같은 중요한 인프라의 경우 여러 가용 영역에서 실행되는 최소 수의 실행 가능한 인스턴스를 프로비저닝하고 가용 영역 중 하나에 장애가 있는 경우 Auto Scaling을 사용하여 로드 증가 또는 내결함성을 처리합니다.

게임 인프라를 여러 리전에 배포하여 가용성을 극대화합니다. 보조 리전에 배포된 간단한 DNS 변경으로 활성화될 수 있는 Aurora 글로벌 데이터베이스 및 중복 인프라와 같은 리전 간 재해 복구 기능은 기본 리전이 손상된 경우 서비스 연속성을 제공할 수 있습니다. 게임 백엔드 서비스가 고가용성을 달성하도록 권장하지만이 권장 사항은 게임 서버에 특히 중요합니다.

예를 들어 멀티플레이어 게임에서는 게임 서버가 플레이어를 위한 게임 세션을 호스팅하는 데 사용되므로 게임 서버의 인프라 용량이 다른 서비스의 용량 요구보다 클 가능성이 높습니다. 많은 게임은 플레이어를 논리적 게임 리전(예: 미국 서부 및 동부)으로 샤딩하도록 선택합니다. 플레이어 경험을 간소화하고 글로벌 인프라를 사용하여 게임을 호스팅하는 작업을 간단하게 수행하려면 게임 서버를 물리적으로 호스팅하는 기본 클라우드 공급자 리전 또는 데이터 센터 위치와 해당 플레이어 게임 리전을 지원하는 게임 서버 인스턴스를 호스팅하는 로컬 영역 또는 자체 데이터 센터와 같은 다른 인프라의 연결을 해제하는 것이 좋습니다.

매치메이킹 서비스를 설계할 때 리전 간에 별도의 소프트웨어 배포를 사용하여 다중 리전 아키텍처를 배포합니다. 매치메이킹 서비스의 리전별 배포가 매치메이킹 요청을 처리했는지에 관계없이 플레이어를 리전의 게임 서버로 라우팅할 수 있도록 게임 서버 인스턴스를 호스팅하는 플릿에서 매치메이킹 서비스 배포를 분리합니다.

매치메이킹 구현에서 지연 시간 및 기타 규칙을 충족하는 게임 서버 리전을 선호하도록 로직을 설계하고, 플릿 용량이 부족하거나 다른 리전 인프라 중단이 있는 경우 플레이어를 다른 리전으로 폴백할 수 있습니다.

구현 단계

- 여러 가용 영역에 게임 인프라를 균일하게 분산하여 고가용성과 복원력을 제공합니다.
- Amazon S3 AWS Lambda, DynamoDB 및 SQS와 같은 관리형를 사용하여 게임 백엔드 서비스 및 상태 저장 구성 요소를 배포하거나 사용자 지정 솔루션을 위한 로드 밸런싱 및 내구성을 구성합니다.
- Aurora 글로벌 데이터베이스 및 기본 물리적 위치와 분리된 논리적 플레이어 대상 리전과 같은 재해 복구 솔루션을 사용하여 중요한 게임 서비스 및 서버에 대한 다중 리전 배포를 구현합니다.

리소스

- 정적 안정성
- Amazon GameLift 게임 세션 대기열 모범 사례
- Amazon GameLift 다중 리전 플릿
- 재해 복구를 위한 Aurora Global Database

변경 관리

GAMEREL02: 수요 변화에 맞게 상태 저장 게임을 어떻게 확장하나요?

플레이어 수요가 시간이 지남에 따라 변동함에 따라 게임 인프라는 이러한 변화하는 요구 사항을 처리하도록 적응형으로 확장할 수 있어야 합니다. 게임의 인기를 미리 예측하기는 어렵지만 플레이어 인구의 변동을 수용할 수 있도록 인프라 용량을 추가 및 제거할 수 있는 아키텍처 접근 방식을 설계합니다.

모범 사례

- GAMEREL02-BP01 활성 플레이어 게임 세션의 상태를 통합하는 조정 전략 구현
- GAMEREL02-BP02 게임에 여러 EC2 인스턴스 유형 사용 지원

GAMEREL02-BP01 활성 플레이어 게임 세션의 상태를 통합하는 조정 전략 구현

활발하게 연결된 플레이어 세션의 상태 저장 특성을 통합하고 게임 플레이를 중단하지 않고 조정 활동을 정상적으로 처리하는 방식으로 게임 인프라를 자동으로 조정하는 솔루션을 구현합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

클라우드에서 게임을 개발할 때 얻을 수 있는 이점 중 하나는 필요에 따라 서버 인프라를 자동으로 조정하여 수요를 충족할 수 있는 탄력성입니다. Amazon EC2 Auto Scaling 정책, EKS Auto Scaling 또는 일반적으로 확장 가능한 웹 애플리케이션에 채택되는 유사한 기술을 사용하여 상태 비저장 또는 비동기 게임 및 백엔드 서비스를 동적으로 확장할 수 있지만, 일반적으로 게임 개발자는 활성 플레이어 세

션의 중단을 차단하기 위해 상태 저장 또는 동기 게임을 확장하기 위한 보다 사용자 지정된 접근 방식을 필요로 합니다.

구현 단계

- 상태 저장 게임의 경우 플레이어 세션의 상태와 사용 가능한 게임 서버 용량을 모니터링하는 데 사용할 수 있는 사용자 지정 지표를 생성하여 Amazon CloudWatch에 사용자 지정 지표로 보고할 수 있습니다. CloudWatch Synthetics와 같은 애플리케이션 모니터링 기능을 사용하여 간단한 가동 및 가동 중지 상태 모니터링에서 감지할 수 없는 기능 장애가 있는지 게임을 확인합니다.
- 사용자 지정 지표를 사용하여 게임 서버 조정 소프트웨어를 AWS Lambda 함수를 사용하는 서버리스 애플리케이션으로 구현하거나 AWS Fargate SDK를 사용하여 게임 서버 빌드를 호스팅하는 EC2 Auto Scaling 그룹의 최소, 최대 및 원하는 용량 설정을 업데이트하는 API 호출 AWS 을 수행하여 전용 게임 서버 인스턴스 플릿을 관리합니다.
- Amazon GameLift를 사용하여 게임 서버를 호스팅하고 out-of-the-box 게임 서버 오토 스케일링 기능을 사용하여이 스케일링 프로세스를 관리합니다.

Amazon GameLift의 자동 조정 기능은 활성 플레이어 세션을 인식하며 플레이어를 적극적으로 호스팅하는 게임 서버 인스턴스의 종료 또는 축소를 차단하도록 구성할 수 있습니다. 자세한 내용은 Amazon CloudWatch를 사용하여 Amazon GameLift 서버 모니터링을 참조하세요.

GAMEREL02-BP02 게임에 여러 EC2 인스턴스 유형 사용 지원

EC2 인스턴스를 사용하여 게임을 호스팅하거나의 EC2 인스턴스에서 호스팅되는 컨테이너를 사용하는 경우 호스팅 전략에서 여러 인스턴스 유형을 AWS 계정사용합니다. 여러 인스턴스 유형을 사용하면 게임 확장 시 사용할 수 있는 컴퓨팅 옵션 수를 늘려 플레이어 증가를 지원하는 서버를 더 추가할 수 있으므로 선호하는 인스턴스 유형을 사용할 수 없는 경우 안정성이 향상됩니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

스팟 인스턴스를 사용하여 게임을 호스팅할 때는 스팟 인스턴스의 가용성이 고객 수요에 따라 변동하므로 여러 인스턴스 유형을 사용하세요.

여러 인스턴스 유형에서 게임을 테스트하여 비용 및 성능 요구 사항을 충족하고 인스턴스 유형의 우선 순위가 지정된 순위를 결정합니다. Amazon EC2 Auto Scaling은 여러 인스턴스 유형 및 크기 사용과 구성의 각 인스턴스 유형에 가중치 할당을 지원하므로 컴퓨팅 옵션의 우선 순위 지정을 구현할 수 있습니다.

Amazon GameLift 관리형 호스팅을 사용하여 게임을 호스팅할 때 게임에 필요한 인스턴스 유형과 해당 인스턴스에서 게임 서버 프로세스를 실행하는 방법(런타임 구성 사용)을 결정합니다. 플릿에 대한 리소스를 선택할 때는 게임 운영 체제, 인스턴스 유형(컴퓨팅 하드웨어), 온디맨드 인스턴스, 스팟 인스턴스 또는 둘 다를 사용할지 여부 등 여러 요인을 고려합니다. Amazon GameLift를 사용한 호스팅 비용은 주로 사용하는 인스턴스 유형에 따라 달라집니다. 자세한 내용은 관리형 플릿의 컴퓨팅 리소스 선택을 참조하세요.

구현 단계

- EC2 또는 컨테이너에서 게임을 호스팅할 때 여러 EC2 인스턴스 유형을 사용하여 신뢰성과 규모 조정 옵션을 개선합니다.
- 우선 순위가 지정된 인스턴스 유형 및 가중치로 Amazon EC2 Auto Scaling 또는 GameLift 플릿을 구성하여 비용과 성능을 최적화합니다.
- 다양한 인스턴스 유형에서 게임을 테스트하여 성능이 요구 사항을 충족하는지 확인하고 그에 따라 호스팅 전략을 조정합니다.

장애 관리

GAMEREL03: 인프라 중단 시 게임 상태를 유지하려면 어떻게 해야 하나요?

게임 인프라는 시간이 지남에 따라 다양한 운영 이벤트를 경험하므로 플레이어 경험의 연속성을 유지하고 인프라 이벤트 중에 게임 상태를 보존하도록 게임의 아키텍처를 설계해야 합니다. 이러한 이벤트를 처리하려면 모니터링, 정상적인 종료 및 상태 지속성 메커니즘을 구현하여 플레이어의 원활한 게임 플레이 경험을 확인합니다.

모범 사례

- GAMEREL03-BP01 게임 서버 중단을 모니터링하고 데이터를 사용하여 호스팅 아키텍처를 개선하여 신뢰성 목표를 달성합니다.
- GAMEREL03-BP02 플레이어 경험에 미치는 영향을 최소화하면서 실패를 처리하기 위해 게임 기능의 느슨한 결합 구현
- GAMEREL03-BP03 시간 경과에 따른 인프라 이벤트를 모니터링하여 플레이어 동작에 미치는 영향 측정

GAMEREL03-BP01 게임 서버 중단을 모니터링하고 데이터를 사용하여 호스팅 아키텍처를 개선하여 신뢰성 목표를 달성합니다.

게임 서버 호스팅 전략을 게임의 신뢰성 요구 사항에 맞게 조정할 수 있도록 게임 서버 지표와 로드 시 지연 시간 증가와 같은 성능 저하가 시간 경과에 따른 플레이어 동작에 미치는 영향을 모니터링합니다. 성능 저하될 게임 서버 인프라는 플레이어에게 영향을 미치는 경우 즉시 서비스에서 제거하거나 서버에 호스팅되는 활성 플레이어 세션이 없는 경우 사전에 교체해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임이 REST APIs로 호스팅되는 시나리오의 경우 시스템 안정성을 기존 웹 애플리케이션 아키텍처처럼 관리할 수 있습니다.이 아키텍처에서는 분산 방식으로 여러 서버에 걸쳐 트래픽을 로드 밸런싱하여 서버 장애 위험을 완화할 수 있습니다.

실시간 동기 게임 플레이의 경우 게임 세션은 일반적으로 가상 머신에서 실행되는 게임 서버 프로세스 또는 게임 서버 인스턴스에서 호스팅됩니다. 게임 플레이 상태를 성능 있는 방식으로 유지하고 연결된 게임 클라이언트에 복제해야 하기 때문입니다. 이 구현은 플레이어의 경험이 게임 세션을 호스팅하는 게임 서버 프로세스의 성능과 신뢰성과 긴밀하게 결합되어 있음을 의미합니다. 이러한 유형의 아키텍처를 사용하면 게임 서버의 신뢰성을 기존 접근 방식보다 더 복잡하게 관리할 수 있습니다.

게임 서버 장애의 영향을 완화하려면 플레이어의 게임 상태를 Amazon Amazon ElastiCache(Redis OSS) 또는 Amazon MemoryDB와 같은 고가용성 캐시 또는 데이터베이스로 비동기식으로 지속적으로 업데이트하도록 게임을 구성합니다. 서버 장애가 발생하면 플레이어의 마지막으로 저장된 게임 상태를 외부 데이터 스토어에서 가져오고 새 게임 서버 인스턴스에서 세션을 복원할 수 있습니다.

그러나이 접근 방식은 외부 상태를 관리하기 위한 추가 비용과 복잡성을 추가하며, 상태 변경이 너무 빈번하고 상당한 규모로 진행되어 성능이 뛰어난 인 메모리 캐시 데이터 스토어를 도입하는 경우 세션을 복원하는 데 유용하기에는 너무 중요한 복제 지연이 발생할 수 있는 빠른 속도의 게임이나 경쟁 게임에는 적합하지 않을 수 있습니다. 이러한 성격의 게임의 경우 최적의 접근 방식은 서버 손실을 수락하고 플레이어를 게임 로비로 다시 보내 다른 세션을 찾거나 자동으로 다른 게임 세션으로 리디렉션하는 것입니다.

나중에 문제를 조사할 수 있도록 서버 중단의 원인에 대한 유용한 로그 데이터를 캡처합니다. Amazon GameLift는 플릿 문제를 디버깅하기 위한 지침을 제공하고 Amazon GameLift 플릿 인스턴스에 원격으로 액세스할 수 있는 기능을 제공합니다.

구현 단계

- 게임 서버 지표의 성능 저하를 모니터링하고 필요에 따라 성능이 저하된 서버를 제거하거나 교체하여 신뢰성을 유지합니다.
- 비동기 게임 상태 업데이트에는 Amazon ElastiCache 또는 MemoryDB를 사용하여 실행 가능한 경우 서버 장애 후 세션 복구를 활성화합니다.
- 플릿 모니터링 및 원격 액세스를 위해 Amazon GameLift와 같은 도구를 활용하여 조사 및 디버깅을 위한 서버 중단에 대한 자세한 로그 데이터를 캡처합니다.

GAMEREL03-BP02 플레이어 경험에 미치는 영향을 최소화하면서 실패를 처리하기 위해 게임 기능의 느슨한 결합 구현

구성 요소 분리란 서버 구성 요소가 최대한 독립적으로 작동하도록 설계하는 개념을 말합니다. 플레이어에게 좋은 게임 내 경험을 제공하려면 데이터를 최대한 최신 상태로 유지해야 하므로 게임의 일부 측면은 분리하기 어렵습니다. 그러나 많은 구성 요소와 게임 작업을 분리할 수 있습니다. 예를 들어 리더보드 및 통계 서비스는 게임 플레이 경험에 중요하지 않으며 이러한 서비스에 대한 읽기 및 쓰기는 게임에서 비동기적으로 수행할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

문제가 감지되면 자동으로 또는 관리자가 비활성화할 수 있는 게임의 기능에 대한 정상적인 성능 저하를 구현하고, 기능에 의존하는 업스트림 서비스를 구성하여 장애를 정상적으로 처리할 수 있도록 합니다. 예를 들어 특정 플레이어 데이터가 게임 클라이언트 내에서 제대로 로드되지 않는 경우이 데이터가 게임 플레이 경험에 중요한지 고려해야 합니다. 그렇지 않은 경우 플레이어의 경험을 방해하지 않고이 실패를 정상적으로 처리하도록 게임 클라이언트를 구성하고 플레이어가 화면을 다시 방문할 때 나중에이 데이터 가져오기를 다시 시도하도록 선택합니다.

제한 시간, 재시도 및 백오프와 같은 로직을 사용하여 오류와 실패를 처리합니다. 제한 시간은 시스템이 불합리하게 장시간 중단되는 것을 방지합니다. 재시도는 일시적 및 무작위 오류의 고가용성을 제공할 수 있습니다.

중요 구성 요소에 느슨하게 결합할 수 있는 중요하지 않은 구성 요소를 정의합니다. 느슨한 결합을 사용하면 한 구성 요소의 장애가 다른 구성 요소로 캐스케이드되지 않으므로 시스템의 복원력이 향상됩니다. 게임 기능에 게임 서버 또는 백엔드에 대한 상태 저장 연결이 필요하지 않은 경우 상태 비저장 프로토콜을 구현하여 동적으로 확장하고 일시적인 장애로부터 복구해야 합니다. HTTP/JSON API를 사

용하여 상태 비저장 프로토콜과 느슨하게 연결할 수 있는 중요하지 않은 구성 요소를 개발합니다. 게임 클라이언트의 네트워크 호출을 비동기식 및 비차단식으로 구현하여 성능이 느린 게임 기능 또는 기타 종속 서비스로 인해 플레이어에게 미치는 영향을 최소화합니다.

느슨한 결합을 통해 복원력을 더욱 개선하려면 비동기적으로 처리할 수 있는 구성 요소 간에 대기열, 스트리밍 또는 주제 기반 시스템과 같은 메시징 서비스를 사용합니다. 이 모델은 즉각적인 응답이 필요하지 않거나 요청이 등록되었다는 확인으로 충분한 상호 작용에 적합합니다. 이 솔루션에는 이벤트를 생성하는 구성 요소와 이벤트를 사용하는 구성 요소가 포함됩니다. 두 구성 요소는 직접적인 point-to-point 상호 작용을 통해 통합되지 않고 내구성 있는 스토리지 또는 대기열 계층과 같은 중간을 통해 통합됩니다. 또한 처리 실패 시 메시지를 보존하여 시스템의 신뢰성을 개선하는 데 도움이 됩니다.

다양한 메시징 서비스가 주문 및 전송 메커니즘과 같은 다양한 특성을 가지고 있으므로 적절한 메시징 메커니즘을 조사하고 선택합니다. 선택한 메시지 시스템이 메시지를 한 번 이상 전송하도록 작업을 멱등성으로 설계합니다. 예를 들어 게임이 플레이어 재생 시간, 통계 또는 기타 관련 데이터를 추적해야 하는 일반적인 게임 사용 사례를 고려하면 플레이어 동시성이 가장 높을 때 쓰기 처리량 사용 사례가 높아질 수 있습니다.

신뢰할 수 있는 아키텍처를 구현하려면 사용 사례에 플레이어가 인식하는 read-after-write 일관성이 필요한지 여부를 고려하세요. 일반적으로 이러한 시나리오는 비동기 처리에 적합하며 요청이 Amazon SQS와 같은 확장 가능하고 내구성 있는 메시지 대기열에 수집되고 Lambda 함수와 같은 소비자 서비스를 사용하여 백엔드 데이터베이스에 배치로 삽입할 수 있는 쓰기 대기열 패턴을 구현하여 달성할 수 있습니다. 이 접근 방식은 플레이어의 게임 클라이언트, 백엔드 웹 및 애플리케이션 서버, 내부 데이터베이스 시스템을 포함한 여러 분산 구성 요소 간의 동기식 통신보다 더 안정적입니다. 또한 필요에 따라 수집 속도를 늦추는 데 쓰기 대기열에서 소비자 처리를 사용할 수 있으므로 최대 쓰기 처리량을 충족하기 위해 백엔드 데이터베이스를 확장할 필요가 없으므로 비용이 절감됩니다.

구현 단계

- 리더보드 및 통계 서비스와 같은 중요하지 않은 구성 요소를 중요한 게임 플레이 기능과 분리하여 비동기 작업을 허용하고 복원력을 개선합니다.
- 제한 시간, 재시도 및 백오프에 대한 로직을 사용하여 중요하지 않은 기능에 대한 정상적인 성능 저하를 구현하고 게임 클라이언트가 플레이어 경험을 방해하지 않고 장애를 처리하는지 확인합니다.
- 구성 요소 간의 비동기 통신을 위해 Amazon SQS와 같은 메시징 시스템을 사용하면 높은 처리량 사용 사례를 확장 가능하고 안정적이며 안정적으로 처리할 수 있습니다.

리소스

- 마이크로서비스 아키텍처를 사용하여 확장성과 신뢰성이 뛰어난 워크로드 구축

- AWS 서버리스 서비스를 사용하여 마이크로서비스 통합
- 마이크로서비스에 대한 비동기식 메시징 이해
- 에서 확장 가능한 게임 개발 패턴 소개 AWS
- 성능 저하 구현

GAMEREL03-BP03 시간 경과에 따른 인프라 이벤트를 모니터링하여 플레이어 동작에 미치는 영향 측정

게임 서버 프로세스, 게임 서버 인스턴스 지표 및 게임 경험 지표를 모니터링하여 문제의 근본 원인을 파악합니다. CPU 및 메모리 모니터링 외에도 EC2 인스턴스의 네트워크 제한과 관련된 네트워크 지표에 대한 모니터링을 설정하여 대역폭 초과, packets-per-second 수 또는 서버 리소스가 프로비저닝되고 있음을 나타낼 수 있는 기타 네트워크 수준 문제와 같은 문제를 알릴 수도 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

CloudWatch Synthetics를 사용하여 로그인할 수 없거나 기타 서비스에 영향을 미치는 문제와 같은 플레이어 경험에 대한 중요한 경로 애플리케이션 기능을 확인합니다. Amazon GameLift를 사용하여 호스팅되는 게임 서버의 경우 다음과 같은 지표를 모니터링하는 것이 좋습니다.

- GameServerInterruptions 및 InstanceInterruptions: 스팟 인스턴스 가용성 제한이 스팟을 사용하여 배포된 게임 서버에 미치는 영향을 이해하는 데 도움이 될 수 있습니다.
- ServerProcessAbnormalTerminations - 게임 서버 프로세스에서 비정상적인 종료를 감지하는 데 사용할 수 있습니다.

게임 서버 신뢰성의 과거 지표 데이터를 유지하는 것이 좋습니다. 이 기록 데이터를 보고 목적으로 사용하고 다른 데이터 세트와 조인하여 잠재적 추세를 파악하고 게임 서버 문제로 인한 시간 경과에 따른 플레이어 동작에 미치는 영향을 평가합니다.

Amazon CloudWatch는 지표를 무기한 보존하지 않으며, 시간이 지남에 따라 지표의 스토리지 해상도가 증가하므로 이러한 지표를 Amazon S3와 같은 비용 효율적인 장기 스토리지로 내보내는 것이 좋습니다. CloudWatch Metric Streams가 CloudWatch 리전에서 자체 S3 버킷으로 지표를 자동으로 전송하도록 구성할 수 있습니다.이 버킷은 S3 Intelligent-Tiering과 같은 스토리지 계층에 장기 저장되고 Amazon Glacier를 사용하여 최종적으로 아카이브될 수 있습니다. Amazon S3에 지표를 배치하면 Amazon Athena와의 대화형 쿼리를 위해 데이터 레이크의 다른 데이터 세트와 쉽게 조인할 수 있습니다.

구현 단계

- 중요한 경로 기능 확인을 위해 Amazon CloudWatch 및 CloudWatch Synthetics를 사용하여 대역폭 및 packet-per-second 수 제한을 포함한 게임 서버, 인스턴스 및 네트워크 지표를 모니터링합니다.
- GameServerInterruptions 및 ServerProcessAbnormalTerminations와 같은 GameLift별 지표를 추적하여 스팟 인스턴스 가용성의 영향을 평가하고 비정상적인 서버 종료를 감지합니다. ServerProcessAbnormalTerminations
- 장기 스토리지를 위해 CloudWatch 지표를 Amazon S3로 내보내고, S3 Intelligent-Tiering 또는 Glacier와 같은 비용 효율적인 계층을 사용하고, Amazon Athena와 같은 도구를 사용하여 추세를 분석합니다.

리소스

- Amazon EC2 인스턴스 수준 네트워크 성능 지표로 새로운 인사이트 발견
- CloudWatch 지표 스트림 - 실시간으로 파트너 및 앱에 AWS 지표 전송

리소스

신뢰성과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

관련 문서:

- 에서 지속적 통합 및 지속적 전달 연습 AWS
- 비동기 작업 대기열 자동 크기 조정
- WorkloadService 아키텍처
- 지터를 사용한 제한 시간, 재시도 및 백오프
- Well-Architected Framework - 신뢰성 원칙
- 안정적인 확장성을 위한 설계
- Amazon Builder 라이브러리
- 멀티플레이어 게임을 위한 대규모 실시간 메시징
- 에서 확장 가능한 게임 개발 패턴 소개 AWS
- 에서 컨테이너화된 마이크로서비스 실행 AWS
- 클라우드에서 웹 애플리케이션 호스팅
- 확장 가능하고 안전한 다중 VPC 네트워크 인프라 구축

관련 비디오:

- re:Invent 2020: Ubisoft:에서 멀티 플랫폼 멀티플레이어 게임 빌드 AWS
- re:Invent 2018: 슈퍼셀 - 모바일 게임 확장
- re:Invent 2019: CAPCOM이 컨테이너, 데이터 및 ML을 사용하여 재미있는 게임을 빌드하는 방법
- re:Invent 2018: 가용성을 유지하면서 Riot Games에서 플레이어 계정 글로벌화
- re:Invent 2020: GameLoft - 가동 중지 시간 없는 데이터 레이크 마이그레이션 심층 분석

관련 교육:

- 게임 서버에 Amazon GameLiftFleetIQ 사용
- Amazon EC2를 사용한 게임 서버 호스팅

성능 효율성

성능 효율성 원칙은 요구 사항을 충족하고 수요 변화 및 기술 발전에 따라 효율성을 유지하기 위해 컴퓨팅 리소스를 효율적으로 사용하는 데 중점을 둡니다.

데이터 기반 접근 방식을 사용하여 고성능 아키텍처를 선택합니다. 상위 수준 설계부터 리소스 유형의 선택 및 구성에 이르기까지 아키텍처에 대한 포괄적인 데이터를 수집합니다. 지속적으로 진화하는 서비스 및 솔루션 세트를 활용하려면 아키텍처 선택을 주기적으로 고려하세요. 지표는 예상 성능과의 편차를 이해하는 데 도움이 되므로 조치를 취할 수 있습니다. 데이터 기반 접근 방식은 아키텍처를 절충하여 성능을 개선하거나 비용을 절감하거나 개발자 경험을 개선하는 데 도움이 됩니다.

중점 영역

- 설계 원칙
- 아키텍처 선택
- 리전 선택
- 반복 개발
- 컴퓨팅 및 하드웨어
- 컴퓨팅 선택
- 데이터 관리
- 네트워킹 및 콘텐츠 전송
- 프로세스 및 문화
- 리소스

설계 원칙

AWS Well-Architected Framework 백서의 설계 원칙 외에도 다음 설계 원칙은 게임의 성능 효율성을 달성할 수 있습니다.

- end-to-end에서 게임 성능 측정: 플레이어의 관점에서 인식되므로 성능을 측정하는 것이 중요합니다. 즉, 게임 클라이언트의 성능, 게임 인프라, 플레이어를 인프라에 연결하는 인터넷 연결을 측정해야 합니다. 이렇게 하면 아키텍처 내에서 성능을 개선할 수 있는 위치를 이해하는 데 도움이 됩니다.
- 아키텍처를 최적화하여 실제 플레이어 경험을 반영하는 지표 개선: 시간이 지남에 따라 아키텍처를 조정하고 발전시킬 때 이러한 개선 및 변경이 플레이어 경험에 어떤 영향을 미칠지 생각해 보세요.

게임 워크로드는 장애의 영향을 견디고 최소화하여 게임 플레이에 대한 광범위한 중단을 차단할 수 있어야 합니다. 서로 크게 의존하지 않는 게임 기능과 시스템을 분리하여 장애의 폭발 반경을 줄이고 플레이어에게 영향을 미치는 문제를 격리해야 합니다.

- 게임 운영을 간소화하고 개발 속도를 높이는 기술 사용: 개발자 효율성을 개선할 수 있는 기술 채택의 우선순위를 정합니다. 사전 프로덕션 개발 단계에서의 운영 오버헤드는 게임 플레이를 개선하는 데 방해가 될 수 있습니다. AWS 또는 AWS 파트너의 관리형 서비스를 활용하면 엔지니어링을 줄일 수 있으므로 게임 개발자는 핵심 게임 루프와 플레이어 경험에 집중할 수 있습니다. 아키텍처 및 성능 요구 사항은 게임 개발 수명 주기 전반에 걸쳐 변경되고 발전할 수 있으며 각 단계에서 기술 장단점을 고려해야 합니다.
- 피크 플레이어 동시성을 충족하고 필요에 따라 동적으로 규모를 조정하도록 인프라 설계: 인프라는 플레이어 수요에 맞게 규모를 조정하도록 설계되어야 합니다. 플레이어 세션 동시성 및 로그인 수와 같은 지표를 사용하여 시스템에 과부하가 걸리기 전에 선제적으로 규모를 조정할 수 있습니다. CPU 및 메모리 소비와 같은 대응 시스템 사용률 지표를 사용하여 시스템에 과부하가 걸린 후 규모를 조정할 수 있습니다. 인프라를 동적으로 확장하면 게임 운영 비용을 줄일 수 있습니다.

아키텍처 선택

GAMEPERF01: 게임 서버에 적합한 호스팅 옵션을 선택하려면 어떻게 해야 하나요?

게임 서버에 적합한 호스팅 옵션을 선택하는 것은 게임 서버 성능의 기본입니다. EC2 인스턴스, 컨테이너 솔루션 또는 완전 관리형 서비스를 사용하기로 결정하는 것은 프로덕션용으로 설계할 때 가장 먼저 내려야 할 결정 중 하나입니다. 각 호스팅 옵션에는 성능 튜닝, 조정, 운영 및 통합에 대한 다양한 기능과 고려 사항이 있습니다.

모범 사례

- GAMEPERF01-BP01 게임 서버 리소스 요구 사항 및 확장성 요구 사항 평가
- GAMEPERF01-BP02 게임 서버 규모 조정을 위한 운영 오버헤드 고려
- GAMEPERF01-BP03 다른 AWS 서비스, 개발 환경, 대상 CPU 아키텍처 및 기능과의 통합 평가

GAMEPERF01-BP01 게임 서버 리소스 요구 사항 및 확장성 요구 사항 평가

확장성 요구 사항에 따라 서버 요구 사항을 평가하여 요구 사항을 충족하고 최적의 성능을 제공하는 호스팅 옵션을 선택하고 있는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임 서버에 적합한 호스팅 옵션을 선택할 때는 다음 요소를 고려하세요.

게임 서버 리소스 요구 사항

게임 서버 프로세스의 CPU, 메모리, 네트워크 및 스토리지 요구 사항을 평가하여 게임이 소비하는 용량을 결정합니다. 네트워킹을 간과하지 마세요. 각 프레임은 플레이어 작업을 수신하고, 게임 상태를 업데이트하고, 플레이어에게 다시 전송하려면 CPU 주기가 필요합니다. 패킷 처리를 오프로드하면 코어 게임 함수에 대한 CPU를 확보할 수 있습니다. 네트워킹은 원활하고 응답성이 뛰어난 게임 플레이의 기반이므로 프로세스 초기에 테스트하면 게임의 기준 성능 프로필이 정의됩니다.

첫 번째 사격 게임은 초당 높은 액션을 가질 수 있으며, CPU는 컴퓨팅 최적화 C 패밀리 인스턴스를 선호할 수 있는 네트워크로 빠르게 이동해야 하는 반면, 턴당 더 많은 CPU 주기 처리를 소비할 수 있는 턴 기반 전략 게임은 플레이어에게 다시 보내기 전에 서버에 게임 상태를 로컬로 저장하고 업데이트하기 위해 R 패밀리 인스턴스의 메모리를 늘려야 할 수 있습니다. 사용률 포화 및 오류(USE) 방법과 같은 데이터 기반 접근 방식을 사용하여 정보에 입각한 아키텍처를 선택할 수 있습니다.

확장성 및 탄력성

각 호스팅 옵션이 얼마나 빠르고 원활하게 성능을 저하시키지 않고 플레이어 수요에 맞게 확장할 수 있는지 평가합니다. 피크 시간에 원활한 게임 경험을 유지하기 위해 게임 워크로드에 필요한 자동화 및 유연성 수준을 고려합니다. 게임 서버는 동일한 인스턴스에 게임 서버 프로세스를 추가하여 사용률을 높여 빠르게 확장할 수 있습니다.이 경우 게임 백엔드는 증가하는 활성 사용자 수와 재생 중인 게임에 따라 더 느리게 확장될 수 있습니다. 플레이어가 게임에 참여할 때까지 기다리는 시간을 최소화하면서 비용을 최소화하려면 플릿이 수요에 따라 확장되어야 합니다. Amazon EC2 스팟 인스턴스 어드바이저를 검토하여 게임 서버 플릿의 비용 효율적인 가용 용량에 대한 인사이트를 얻으세요.

구현 단계

- FPS 게임의 높은 네트워크 처리량 또는 턴 기반 전략 게임의 메모리 최적화와 같은 게임별 성능 요구 사항을 고려하여 CPU, 메모리, 네트워크 및 스토리지에 대한 게임 서버 리소스 요구 사항을 평가하여 적절한 인스턴스 유형을 선택합니다.
- USE 메서드와 같은 프레임워크를 사용하여 성능 데이터를 분석하여 컨테이너, 인스턴스, 베어 메탈, 관리형 서비스와 같은 다양한 호스팅 옵션을 비교합니다. 이러한 인사이트를 사용하여 시스템 아키텍처에 대해 더 나은 결정을 내릴 수 있습니다.
- 확장성과 탄력성을 위해 플릿을 설계하고 EC2 스팟 인스턴스 어드바이저와 같은 도구를 활용하여 비용을 최적화하는 동시에 피크 시간 동안 플레이어 수요에 맞게 빠르게 조정할 수 있습니다.

GAMEPERF01-BP02 게임 서버 규모 조정을 위한 운영 오버헤드 고려

각 호스팅 옵션과 관련된 관리 및 운영 오버헤드를 고려합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

운영 오버헤드

EC2 또는 컨테이너의 자체 호스팅 솔루션은 더 많은 제어를 제공할 수 있지만 더 많은 관리가 필요합니다. ECS 또는 EKS와 같은 컨테이너 오케스트레이터는 컨테이너화된 서버의 시작 시간을 줄이는 동시에 네트워킹 복잡성과 유지 관리 오케스트레이션 오버헤드를 높일 수 있습니다.

예를 들어 EKS 관리형 노드 그룹은 게임 서버의 프로비저닝 및 수명 주기 관리를 자동화할 수 있지만 노드를 종료할 때 포드 중단 예산을 준수하지 않습니다. 게임을 안전하게 완료하는 데 15분 이상의 종료 기간이 필요한 경우 수명 주기 후크를 생성하거나 사용자 지정 컨트롤러가 있는 자체 관리형 노드를 고려하여 게임 중단을 차단해야 할 수 있습니다.

Amazon Game Lift와 같은 관리형 서비스는 대부분의 운영 오버헤드를 처리하지만 낮은 수준의 네트워킹 및 보안 구성에 대한 특수 요구 사항에 대한 가시성과 제어 수준을 줄일 수 있습니다. 게임 서버 솔루션을 선택하는 것은 게임 서버 성능을 조정하고 동작을 조정하는 데 필요한 사용자 지정, 제어 및 책임 수준 간의 장단점입니다.

구현 단계

- 호스팅 옵션에 대한 운영 오버헤드를 평가하고 EC2, ECS 또는 EKS와 같은 자체 호스팅 솔루션과 Amazon Game Lift와 같은 관리형 서비스 간의 제어 및 관리 작업의 균형을 조정합니다.
- 자동화에 EKS 관리형 노드 그룹을 사용하지만 게임 서버에 기본값보다 긴 종료 기간이 필요한 경우 수명 주기 후크 또는 사용자 지정 컨트롤러를 구현합니다.
- 게임 서버 솔루션을 선택할 때 사용자 지정, 가시성 및 운영 책임 간의 장단점을 평가합니다.

GAMEPERF01-BP03 다른 AWS 서비스, 개발 환경, 대상 CPU 아키텍처 및 기능과의 통합 평가

각 호스팅 옵션이 데이터베이스, 분석 또는 콘텐츠 전송 서비스와 같이 게임이 의존하는 다른 AWS 서비스와 얼마나 잘 통합되는지 평가합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

다른 AWS 서비스와의 통합

서비스 간 원활한 통합은 성능 모니터링 개선, 게임 구성 요소, 게임 서버, 게임 백엔드 서비스 및 관찰성 솔루션 간의 효율적인 보안 데이터 전송과 같은 운영상의 이점을 제공합니다.

예를 들어 라이브 게임의 트래픽 이동 조정은 복잡할 수 있습니다. Amazon Route 53를 사용하면 DNS 레코드를 최신 상태로 유지하여 조정된 트래픽 전환을 간소화할 수 있습니다. AWS Global Accelerator 트래픽 다이얼을 사용하면 트래픽의 일부를 다른 리전으로 보내고 유지 관리 중에 게임을 계속 실행할 수 있습니다.

개발 환경 및 도구

각 아키텍처 옵션에서 지원하는 개발 도구, 프레임워크 및 환경을 고려합니다. 선택한 옵션이 게임 개발 솔루션 및 프로그래밍 언어와 일치하는지 확인합니다. 게임 서버 성능을 최적화하고 유지하는 팀의 능력에 영향을 미칠 수 있기 때문입니다. 모바일, 콘솔 및 PC에서 게임을 제공하면 도구 및 테스트 복잡성이 증가합니다. 교차 시스템 지원은 중앙 집중식 서비스가 여러 제목에서 개발 모범 사례를 표준화할 수 있는 다중 게임 스튜디오에 특히 중요합니다.

대상 CPU 아키텍처 및 기능

게임 엔진 및 게임 서버 프로세스의 성능 프로파일과 사용 가능한 ARM 지원 수준을 고려합니다. ARM 기반 Graviton 또는 x86 기반 AMD64 프로세서의 향상된 가격 대비 성능을 활용할 수 있는지 평가합니다. AES-NI 암호화, AVX 또는 Turbo Boost와 같은 Intel 기능을 사용해야 합니까? 전용 호스트 유형을 검토하여 단일 및 다중 소켓 인스턴스 패밀리를 식별합니다. 다중 소켓 인스턴스 패밀리를 사용하는 경우 게임 서버 프로세스에서 NUMA 고정 및 L3 캐시 공유를 고려하세요. C 상태 및 P 상태 구성을 사용하면 주파수 클럭을 튜닝하고 절전 레벨을 줄여 게임에 가장 적합한 성능을 얻을 수 있습니다.

구현 단계

- 성능 모니터링을 간소화하고 AWS , 데이터 전송을 보호하며, 수동 운영 작업을 줄이는 데 도움이 되도록 AWS Secrets Manager, ACM 등과 같은 서비스와 원활하게 통합되는 호스팅 옵션을 선택합니다.
- 호스팅 옵션과 개발 환경, 프레임워크 및 프로그래밍 언어 간의 호환성을 확인하여 서버 성능을 효과적으로 최적화하고 유지합니다.
- 가격 대비 성능에는 Graviton을, AES-NI, AVX, Turbo Boost와 같은 특정 기능에는 x86을 활용하여 CPU 아키텍처 요구 사항을 평가하고, NUMA 고정 및 C 상태/P 상태 튜닝을 통해 서버 성능을 최적화합니다.

리전 선택

GAMEPERF02: 게임 인프라를 호스팅할 지리적 리전을 어떻게 결정하나요?

게임 인프라에 적합한 위치를 선택하면 플레이어와 백엔드의 네트워킹 성능이 향상될 수 있습니다. 플레이어 기반이 어디에서 연결되고 커뮤니티 또는 서버가 구축되는 방식을 고려하는 것은 지리적 리전의 장기적인 성장과 지속 가능성에 중요합니다. 분리된 게임 서버 인프라와 백엔드 서비스를 배포하면 여러 리전, 로컬 영역 및 Outpost를 사용하여 게임을 호스팅함으로써 전반적인 운영 효율성을 높이고 복원력을 개선할 수 있습니다.

모범 사례

- GAMEPERF02-BP01 플레이어 근처에 있는 홈 리전 선택
- GAMEPERF02-BP02 플레이어와 가까운 곳에 지연 시간에 민감한 게임 인프라를 배치하여 성능을 개선하도록 지원하는 접근 방식 설계

GAMEPERF02-BP01 플레이어 근처에 있는 홈 리전 선택

초기 게임 출시의 경우 플레이어가 게임을 사용할 수 있을 것으로 예상되는 게시 팀과 출시 전 마케팅 및 광고 작업에 집중하는 게시 팀 등 비즈니스 이해관계자와의 논의를 기반으로 인프라를 배포할 위치를 결정해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

또한 비즈니스 이해관계자에게는 플레이어 수신 및 실행 가능성을 더 잘 이해하기 위해 수요를 촉진할 수 있는 메커니즘이 있어야 합니다. 예를 들어 이러한 팀에는 게임 사전 주문, 마케팅 이벤트 및 캠페인, 출시 전에 플레이어가 관심을 등록할 수 있는 공개 이메일 목록, 출시 시 게임 플레이어가 가장 많을 수 있는 위치를 결정하기 위한 관련 신호를 설정하는 기타 접근 방식과 같은 메커니즘이 있습니다. 게임은 재생 테스트 및 소프트 런치 단계를 포함하는 리전별 롤아웃 전략을 사용하여 리전별 플레이어 수요를 결정할 수도 있습니다.

플레이어 기반과 개발자 근처에 있고 게임을 호스팅하는 데 필요한 AWS 서비스와 기능을 갖춘 홈 리전을 선택합니다. 홈 RSegion은 게임 백엔드 서비스가 실행되는 위치이며 게임 서버를 실행할 수도 있습니다. 지원되는 서비스, 엣지 로케이션에 대한 연결, 장애 조치 리전에 대한 근접성, 가용 영역 수를 기반으로 홈 리전을 평가합니다. 로컬 영역을 사용하는 경우 상위 리전이 때때로 다른 지리적 영역

에 있다고 가정합니다. 예를 들어, 상파울루 sa-east-1에 지리적으로 더 가깝더라도 칠레 로컬 영역 us-east-1-scl-1a에는 버지니아 북부 us-east-1이 상위 리전으로 있습니다.

구현 단계

- 사전 주문, 마케팅 캠페인, 관심 등록과 같은 출시 전 활동의 플레이어 수요 신호를 기반으로 배포 리전을 식별합니다.
- 기본 플레이어 기반 및 개발자와 가까운 홈 리전을 선택하여 필요한 AWS 서비스, 엣지 로케이션 및 장애 조치 리전을 지원하는지 확인합니다.
- 상위 리전이 로컬 영역의 위치와 지리적으로 다를 수 있음을 고려하여 로컬 영역을 신중하게 평가합니다.

GAMEPERF02-BP02 플레이어와 가까운 곳에 지연 시간에 민감한 게임 인프라를 배치하여 성능을 개선하도록 지원하는 접근 방식 설계

게임 서버와 같은 지연 시간에 민감한 인프라를 별도로 배치하면 긴 네트워크 경로의 영향을 최소화할 수 있습니다. 반복 가능한 배포를 통해 플레이어에게 더 나은 성능을 제공하는 여러 위치를 간단하게 유지할 수 있습니다. Ping은 게임 UI에 표시되는 일반적인 지표이며 낮은 ping은 차별화 기능일 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임을 처음 시작할 때 플레이어 기반에 대한 정보가 충분하지 않아 게임 플레이에 가장 관심이 있는 플레이어와 가장 가까운 인프라를 배포할 수 있는 위치를 적절하게 알 수 없습니다. 이는 일반적인 과제이므로 플레이어에게 더 가까운 위치에 서버를 배포하기 위해 호스팅 배치 전략을 신속하게 조정할 수 있는 아키텍처를 설계하여이 시나리오에 대비해야 합니다. 게임 개발자는 일반적으로 게임 인프라 배포를 시작 후 반복 분석으로 정기적으로 평가하여 반복적인 접근 방식을 통해 시간이 지남에 따라 개선에 점진적으로 투자합니다.

가장 좋은 방법은 AWS CloudFormation 또는 Hashicorp의 Terraform과 같은 infrastructure-as-code 템플릿을 사용하여 중요한 게임 서비스를 시작하는 데 필요한 VPCs, 서브넷 구성 및 종속성과 같은 인프라를 구성하여 이러한 템플릿을 참조하고, 필요한 경우 신속하게 사용자 지정하고, 플레이어를 지원하기 위해 추가 인프라가 필요한 위치에 배포할 수 있도록 하는 것입니다.

또한 향후 확장을 위해 현재 배포 전략을 어떻게 발전시킬 수 있는지 이해해야 합니다. IaC 템플릿은 반복 가능하지만 네트워크 계획을 대체하지는 않습니다. IPAM은 VPCs 관리합니다. 서브넷 크기 조정, 가용 영역 선택, IP 인벤토리 및 교차 계정 가용 영역 정렬. 네트워크는 고려해야 할 중요하며 변경될 경

우 플레이어에게 방해가 될 수 있습니다. 여러 지리적 위치에 배포된 게임 서버는 게임 백엔드에 연결되며, 프라이빗 연결을 지원하기 위해 추가 구성이 필요할 수 있는 단일 또는 여러 홈 리전에서 호스팅하는 것이 더 일반적입니다. 게임 요구 사항이 진화하거나 플레이어 요구 사항이 변경될 때 게임 호스팅 전략을 변경할 수 있도록 시간이 지남에 따라 이러한 고려 사항을 지속적으로 평가해야 합니다.

게임에 사용할 게임 호스팅 위치 수를 결정할 때는 다음 요소를 고려하세요.

- 플레이어 경험 개선의 품질: 게임 호스팅 위치를 추가하여 플레이어 경험을 얼마나 개선할 수 있나요? 이렇게 하면 얻을 수 있는 점진적인 성능 향상은 무엇입니까? 이 성능 개선을 어떻게 측정하시겠습니까?
- 우선 순위를 지정할 플레이어 모집단: 게임 호스팅 위치를 추가할 경우 경험을 개선할 수 있는 플레이어는 몇 명입니까? 어떤 플레이어 모집단 또는 지리적 위치를 우선시하시겠습니까?
- 변화의 다운스트림 영향: 게임 호스팅 전략을 변경하면 플레이어의 매치메이킹 대기 시간에 어떤 영향을 미치나요? 플레이어 풀의 매치 크기, 스킬 밸런스 또는 플레이어 수가 게임 호스팅 위치 전략 변경을 수용할 수 있나요? 더 많은 위치를 지원하면 플레이어 풀이 조각화되어 비용과 복잡성이 증가할 수 있습니다.

이러한 각 고려 사항은 게임 호스팅 위치를 추가하거나 제거하는 위치를 결정할 때 평가해야 합니다. 예를 들어 성능이 가장 낮은 게임 플레이 경험을 제공하는 지리적 위치의 플레이어 또는 가장 음성적인 공개 피드백을 표현하는 플레이어의 경험을 개선하는 데 우선순위를 두도록 선택할 수 있습니다. 플레이어 수익 창출을 우선순위에 반영하도록 선택할 수도 있습니다. 예를 들어, 게임에 상당한 수익을 창출하거나 성능 개선을 도입할 경우 증분 수익을 창출할 가능성이 있는 지리적 위치에서 플레이어의 경험을 개선하는 데 집중할 수 있습니다.

에서 인프라를 호스팅하는 것 외에도 AWS 리전의 확장인 로컬 영역을 사용하여 게임 서버와 플레이어 AWS 리전와 더 가까운 음성 채팅 서버와 같은 기타 지연 시간에 민감한 애플리케이션을 호스팅할 수 있습니다. 또한 로컬 영역에서 게임 개발 인프라를 실행하여 게임 개발 팀의 경험을 개선하도록 선택할 수도 있습니다. 예를 들어 로컬 영역을 사용하여 게임 개발자와 더 가까운 곳에서 자체 관리형 소스 제어 서버의 복제본을 호스팅하는 등의 사용 사례를 해결하고, 온프레미스에서 인프라를 호스팅할 필요 없이 개발 스튜디오 근처의 하나 이상의 로컬 영역에 배포된 Amazon EC2 인스턴스, EBS 볼륨 및 Amazon FSx 파일 시스템을 사용하여 사용자에게 게임 개발 가상 워크스테이션 및 콘텐츠 스토리지를 제공할 수 있습니다.

동일한 지리적 영역에서 리전 또는 로컬 영역을 사용할 수 없는 경우 Outpost를 선택하는 것이 좋습니다. 게임 서버가 시스템 신뢰성을 백엔드할 수 있도록 데이터 센터와의 연결을 고려해야 AWS 합니다. AWS Outposts Outpost 서버는 게임을 실행할 때마다 일관된 배포 모델을 생성하는 데 도움이 되도록 동일한 서비스 및 APIs를 사용하여 AWS 데이터 센터에서 실행되도록 특별히 설계되었습니다. 여러 랙

을 논리적 Outpost에 결합할 수 있으며 인프라를 공유할 수 있습니다 AWS 계정. 하드웨어 수명 주기는 에서 관리 AWS 하며 리드 타임은 3개월로 짧을 수 있습니다.

컨테이너를 사용하여 게임을 구축하고 자체 온프레미스 인프라에 배포할 수 있는 오픈 소스 소프트웨어를 사용하여 하이브리드 배포 아키텍처를 유연하게 채택하려는 경우 또는 AWS Outposts 로컬 영역의 대안으로 ECS Anywhere 또는 EKS Anywhere를 사용할 수 있습니다. Amazon GameLift로를 호스팅하는 경우 Amazon GameLift Anywhere를 사용하여 개발 프로세스의 속도를 높일 수 있는 로컬 하드웨어에서 서버 빌드를 실행할 수 있으므로 로컬 영역을 사용하거나 자체 금속을 플릿의 일부로 등록할 수 있습니다.

구현 단계

- 반복 가능한 배포를 위해 AWS CloudFormation 또는 Terraform과 같은 infrastructure-as-code 도구를 사용하면 플레이어 요구 사항에 따라 게임 호스팅 위치를 빠르게 사용자 지정하고 확장할 수 있습니다.
- 플레이어 경험 개선 사항, 플레이어 인구 우선 순위, 게임 호스팅 위치를 추가하거나 제거할 때 매치메이킹 시간과 같은 다운스트림 영향을 평가합니다.
- AWS Local Zones, Outposts 또는 ECS Anywhere, EKS Anywhere 또는 GameLift Anywhere와 같은 하이브리드 옵션을 사용하여 지연 시간에 민감한 인프라를 최적화하고 다양한 배포 요구 사항을 지원합니다.

반복 개발

GAMEPERF03: 반복적인 개발 성능 효율성을 위해 Amazon GameLift를 어떻게 사용할 수 있나요?

Amazon GameLift는 로컬 테스트 환경에서 게임의 개발 및 성능 테스트를 위한 end-to-end 워크플로를 제공합니다.

모범 사례

- GAMEPERF03-BP01 Amazon GameLift Anywhere 및 GameLift 테스트 도구 키트 사용
- GAMEPERF03-BP02 게임 서버의 성능 및 확장성 테스트
- GAMEPERF03-BP03 GameLift 컨테이너의 리소스 사용률 최적화

GAMEPERF03-BP01 Amazon GameLift Anywhere 및 GameLift 테스트 도구 키트 사용

반복 개발 프로세스를 통해 성능 효율성을 높이려면 Amazon GameLift Testing Toolkit과 함께 Amazon GameLift Anywhere를 활용하여 포괄적인 테스트 환경을 구축하세요.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

이 접근 방식을 사용하면 빠른 반복, 효율적인 데이터 수집 및 상세한 성능 분석이 가능합니다. 주요 단계는 다음과 같습니다.

테스트 환경 생성

Amazon GameLift Anywhere를 사용하여 로컬 또는 클라우드 기반 테스트 환경을 설정합니다. 이 설정을 사용하면 각 게임 서버 빌드 반복을 관리형 플릿에 업로드할 필요가 없으므로 활성화 시간이 단축됩니다.

Amazon GameLift 테스트 도구 키트 통합

Amazon GameLift 테스트 도구 키트를 개발 워크플로에 통합합니다. 도구 키트는 Amazon GameLift 인프라를 시각화하고, 가상 플레이어를 시작하고, FlexMatch 시뮬레이터를 사용하여 FlexMatch 규칙 세트를 반복하는 스크립트, 도구 및 라이브러리를 제공합니다. Amazon GameLift 리소스의 통합 및 관리를 간소화하여 일반적인 작업을 자동화하고 성능 분석에 필요한 데이터를 수집할 수 있습니다.

빠른 빌드 및 테스트 주기

새 빌드로 테스트 플릿을 빠르게 업데이트하고 시작하고 테스트를 시작합니다. 이를 통해 빠른 build-test-repeat 주기를 용이하게 하여 개발자가 멀티플레이어 상호 작용을 포함하여 게임 플레이어 경험의 다양한 측면을 검증할 수 있습니다.

포괄적인 테스트

Amazon GameLift 서버 SDK, 백엔드 서비스 상호 작용, 매치메이킹 구성 및 기타 GameLift 호스팅 기능과의 게임 서버 통합을 테스트합니다. GameLift 테스트 도구 키트를 활용하여 테스트를 자동화하고 자세한 성능 지표를 수집하여 게임 구성 요소가 함께 원활하게 작동하는지 확인합니다.

성능 데이터 분석

GameLift Testing Toolkit에서 수집한 데이터를 사용하여 성능 병목 현상을 분석하고 게임 서버를 최적화합니다. 이 툴킷은 주요 지표를 추적하고, 문제를 식별하고, 성능 효율성을 개선하기 위해 데이터 기반 결정을 내리는 데 도움이 됩니다.

Amazon GameLift Anywhere와 GameLift Testing Toolkit을 반복 개발 프로세스에 통합하여 신속한 테스트, 포괄적인 통합 검사 및 세부 성능 분석을 통해 성능 효율성을 크게 향상시킬 수 있습니다.

구현 단계

- Amazon GameLift Anywhere를 사용하여 테스트 환경을 생성하여 게임 서버 빌드의 활성화 시간을 줄이고 빠른 반복을 활성화합니다.
- Amazon GameLift Testing Toolkit을 통합하여 테스트 작업을 자동화하고, 플레이어를 시뮬레이션하고, 개발 중에 FlexMatch 구성을 검증합니다.
- GameLift 테스트 도구 키트를 사용하여 성능 데이터를 수집하고 분석하여 병목 현상을 식별하고, 게임 서버를 최적화하고, 성능 효율성을 개선합니다.

GAMEPERF03-BP02 게임 서버의 성능 및 확장성 테스트

게임 서버의 성능과 확장성을 테스트하려면 Amazon GameLift 기능과 GameLift Testing Toolkit을 사용하여 강력한 테스트 프레임워크를 구현합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

주요 사례는 다음과 같습니다.

반복 테스트

Amazon GameLift Anywhere 플릿을 사용하여 게임 구성 요소를 반복적으로 빌드하고 테스트할 수 있는 클라우드 기반 호스팅 환경을 생성합니다. 이 환경은 실제 호스팅 조건을 미러링하여 사실적인 성능 및 확장성 테스트를 가능하게 해야 합니다.

게임 서버 통합 테스트

AWS CLI 또는 GameLift Testing Toolkit을 사용하여 새 게임 세션 시작 및 게임 세션 이벤트 추적을 포함하여 게임 서버와 Amazon GameLift 서버 SDK의 통합을 테스트합니다. 이렇게 하면 게임 서버가 GameLift 환경 내에서 올바르게 작동하는지 확인할 수 있습니다.

GameLift 테스트 도구 키트를 사용하여 테스트를 자동화하고 자세한 성능 지표를 수집합니다. 도구 키트를 사용하면 GameLift 인프라를 시각화하고, 로드 테스트를 위해 가상 플레이어를 시작하고, FlexMatch 시뮬레이터를 사용하여 FlexMatch 규칙 세트를 반복할 수 있습니다. 이는 서버 인프라를 스트레스 테스트하기 위해 수많은 동시 게임 세션을 생성하여 플레이어 세션을 시뮬레이션하는 ECS Fargate 작업을 조정하는 데 특히 유용합니다.

확장성 테스트

게임 세션 대기열 설계, 다중 위치 플릿, 스팟 및 온디맨드 플릿, 여러 인스턴스 유형을 실험합니다. 게임 세션 배치 옵션, 지연 시간 정책 및 플릿 우선 순위 설정을 테스트합니다. 플레이어 수요에 맞게 용량 조정을 구성하고 시스템이 다양한 조건에서 예상 로드를 처리할 수 있는지 확인합니다.

구현 단계

- Amazon GameLift Anywhere를 사용하여 반복 성능 및 확장성 테스트를 위한 현실적인 테스트 환경을 설정합니다.
- GameLift 서버 SDK와의 게임 서버 통합을 테스트하여 GameLift 환경 내에서 올바른 세션 관리 및 이벤트 추적을 용이하게 합니다.
- GameLift 테스트 도구 키트를 사용하여 확장성 테스트를 수행하고, 플레이어 로드를 시뮬레이션하고, 세션 대기열을 테스트하고, 플릿 조정, 지연 시간 정책 및 우선 순위 설정을 검증합니다.

GAMEPERF03-BP03 GameLift 컨테이너의 리소스 사용률 최적화

GameLift 컨테이너의 리소스 사용률을 최적화하려면 컨테이너 플릿을 효과적으로 설계하고 정확한 리소스 제한을 설정합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

주요 지침은 다음과 같습니다.

- 컨테이너 그룹 설계: 소프트웨어를 컨테이너 그룹으로 구성합니다. 기본 컨테이너는 게임 서버 애플리케이션과 Amazon GameLift 에이전트를 번들링해야 합니다. 추가 소프트웨어에 사이드카 컨테이너를 사용하여 종속성을 관리하고 메모리 및 CPU 사용량에 대한 컨테이너별 제한을 설정합니다.
- 리소스 제한 설정: 각 컨테이너 그룹에 대해 필요한 메모리 및 CPU 리소스를 결정합니다. 개별 컨테이너에 대해 선택적 제한을 설정하여 리소스가 예약되어 있는지 확인하지만 추가 리소스를 사용할 수 있는 경우 이러한 제한을 초과할 수도 있습니다. 이렇게 하면 리소스 경합과 잠재적인 컨테이너 장애를 방지하는 데 도움이 됩니다.

- 데몬 컨테이너 그룹: 기본 컨테이너 그룹으로 확장할 필요가 없는 백그라운드 또는 모니터링 프로세스에 데몬 컨테이너 그룹을 사용하는 것이 좋습니다. 이렇게 하면 기본 게임 서버 프로세스에 영향을 주지 않고 필수 백그라운드 작업이 효율적으로 처리되는지 확인할 수 있습니다.

구현 단계

- 게임 서버 및 GameLift 에이전트용 기본 컨테이너와 특정 메모리 및 CPU 제한으로 종속성을 관리하기 위한 사이드카를 사용하여 컨테이너 그룹을 설계합니다.
- 경합을 방지하기 위해 제어된 리소스 사용을 허용하면서 필요한 리소스를 예약하려면 각 컨테이너 그룹에 대한 리소스 제한을 설정합니다.
- 백그라운드 또는 모니터링 작업에 데몬 컨테이너 그룹을 사용하여 기본 게임 서버 프로세스에 영향을 주지 않고 효율적으로 작동하는지 확인합니다.

컴퓨팅 및 하드웨어

GAMEPERF04: 게임 세션이 동일한 게임 서버 인스턴스에서 실행되는 플레이어에게 영향을 미치지 않도록 차단하려면 어떻게 해야 하나요?

게임 서버가 실행된 후에는 성능을 모니터링하여 리소스 사용률 AWS, 기본 컴퓨팅 또는 포화도에 관계없이 고품질 플레이어 경험을 제공해야 합니다.

모범 사례

- GAMEPERF04-BP01 게임 서버 프로세스를 모니터링하여 문제 감지
- GAMEPERF04-BP02 시뮬레이션된 실제 게임 플레이 시나리오로 게임 서버 성능 테스트

GAMEPERF04-BP01 게임 서버 프로세스를 모니터링하여 문제 감지

인스턴스당 여러 게임 서버 프로세스를 실행하여 게임 서버 인스턴스의 리소스를 효율적으로 활용할 수 있습니다. 그렇다면 게임 세션을 호스팅하는 개별 게임 서버 프로세스가 동일한 인스턴스에서 호스팅되는 다른 게임 세션에 부정적인 영향을 미치지 않도록 아키텍처를 설계하십시오. 지표를 사용하여 게임 배치 및 게임 모드 유형이 게임 서버 인스턴스의 성능에 미치는 영향을 이해합니다. 낮은 로드(로비, 상점 또는 단일 플레이어 자습서)와 높은 로드(순위, 멀티플레이어 또는 높은 스킬 게임 플레이) 프로세스를 혼합하여 게임 서버 인스턴스를 핫 스팟하지 않도록 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

ping 시간 및 지터, 프레임 드롭, API 응답 시간, 오류 및 성공적인 게임 루프 완료에 대한 원격 측정을 수집하여 클라이언트 측 및 서버 측 지표를 통해 플레이어 경험을 모니터링합니다. 이러한 이벤트의 타임스탬프를 플레이어 지원 문제 및 서버 로그와 연관시켜 성능 병목 현상을 식별합니다. Dtrace, ftrace, uperf 및 eBPF와 같은 도구를 사용하여 시스템 성능을 심층적으로 조사하고 분석할 수 있습니다.

게임 서버 인스턴스에 사용할 수 있는 제한된 리소스에 대한 모니터링을 구현하여 개별 게임 서버 프로세스가 미리 결정된 리소스 예산 임계값을 위반할 때 알림을 생성할 수 있습니다. 임계값을 위반하면 게임 서버 엔지니어가 이러한 동작을 조사할 수 있도록 관련 시스템 및 게임 서버 로그 아웃을 중앙 로깅 솔루션과 같은 내구성 있는 스토리지로 덤프하도록 게임 서버 소프트웨어를 구성할 수 있습니다. 또한 게임 서버 인스턴스에 대한 전체 지표 외에도 이러한 개별 게임 서버 프로세스를 모니터링할 수 있도록 인스턴스에서 실행되는 각 게임 서버 프로세스의 지표를 보고하도록 게임 서버 인스턴스를 구성해야 합니다.

예를 들어 GameLift는 게임 세션 모니터링을 위한 지표를 제공합니다.이 지표는 게임 서버 인스턴스에서 구성할 수 있는 Amazon CloudWatch 에이전트를 사용하여 수집된 사용자 지정 게임별 지표 및 로그로 보강할 수 있습니다. 지표는 CloudWatch에서 보거나 Single Sign-On과 통합된 Amazon Managed Grafana와 같은 다른 도구로 내보내 Management Console에 액세스할 수 없는 사용자가 지표에 쉽게 액세스할 수 있습니다. 개별 게임 세션 로그 보기에 대한 지원도 제공하는 Amazon GameLift를 사용하여 로그 및 지표를 관리하는 다음 모범 사례를 참조하세요. https://docs.aws.amazon.com/gamelift/latest/apireference/API_GetGameSessionLogUrl.html

구현 단계

- 인스턴스당 여러 게임 서버 프로세스를 실행하고 로드가 적은 게임 모드와 높은 게임 모드를 혼합하여 핫 스팟을 방지하고 균형 잡힌 리소스 사용률을 확인합니다.
- ping, 지터, 프레임 드롭, API 응답 시간과 같은 클라이언트 측 및 서버 측 지표를 모니터링하고 플레이어가 보고한 서버 로그 및 문제와 상호 연관시켜 병목 현상을 식별합니다.
- CloudWatch 및 Amazon Managed Grafana와 같은 도구를 사용하여 각 게임 서버 프로세스에 대한 리소스 모니터링을 구성하고, 임계값 위반에 대한 알림을 생성하고, 분석을 위해 로그를 내구성 있는 스토리지에 저장합니다.

GAMEPERF04-BP02 시뮬레이션된 실제 게임 플레이 시나리오로 게임 서버 성능 테스트

성능 테스트를 수행하고 다양한 게임 플레이 시나리오를 평가하여 게임 서버 프로세스가 EC2 인스턴스 메모리, CPU 및 네트워크 대역폭과 같은 고정 리소스의 사용률을 적절하게 처리하는지 확인합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

플레이어의 일반적인 게임 플레이 경로와 동작을 미러링할 수 있는 봇을 사용하여 시뮬레이션된 게임 플레이 테스트를 생성하면 게임 서버 프로세스가 다양한 사용 시나리오에서 이를 처리하는 방법을 결정할 수 있습니다. 예를 들어 게임 클라이언트 시뮬레이션 또는 게임 클라이언트 빌드를 실행하여 게임 플레이 시나리오를 생성하도록 사용자 지정할 수 있는 의 분산 로드 테스트 AWS와 같은 솔루션을 구현할 수 있습니다. 내부 플레이 테스트를 실행하고 QA 팀을 사용하여 게임의 다양한 기능을 스트레스 테스트하여 게임이 최적의 성능을 발휘하도록 설계되었다는 확신을 얻을 수 있습니다.는 여러 디바이스 유형에서 iOS, Android 및 브라우저 게임에 대한 모바일 및 웹 테스트를 수행하는 데 사용할 AWS Device Farm 수 있습니다.

구현 단계

- 일반적인 플레이어 동작을 시뮬레이션하는 봇을 사용하여 성능 테스트를 수행하여 다양한 시나리오에서 게임 서버 리소스 사용률을 평가합니다.
- 의 분산 로드 테스트와 같은 솔루션을 사용하여 스트레스 테스트를 위한 게임 플레이 시나리오를 사용자 지정하고 시뮬레이션 AWS 할 수 있습니다.
- 다양한 디바이스에서 모바일 및 브라우저 게임 테스트를 AWS Device Farm 위해 내부 플레이테스트를 수행하고와 같은 도구를 사용합니다.

컴퓨팅 선택

GAMEPERF05: 게임에 적합한 컴퓨팅 솔루션을 어떻게 선택하나요?

컴퓨팅 성능은 인스턴스 크기와 패밀리에 따라 다릅니다. 별도의 용량 풀에 있는 여러 컴퓨팅 옵션을 사용하는 것이 좋습니다. 성능에 대한 선호도를 제공하지만 용량 부족 오류를 방지하기에 충분한 다양성을 포함하는 플릿 구성 전략을 개발합니다.

모범 사례

- GAMEPERF05-BP01 여러 컴퓨팅 유형에서 게임 성능 벤치마크
- GAMEPERF05-BP02 non-latency-sensitive 컴퓨팅 작업을 비동기 워크플로로 이동

GAMEPERF05-BP01 여러 컴퓨팅 유형에서 게임 성능 벤치마크

게임 서버 워크로드의 경우 게임 서버를 호스팅하기 위한 최적의 컴퓨팅 솔루션을 식별하는 단일 접근 방식은 없습니다. 게임 서버를 벤치마킹하기 위한 일반적인 전략은 컴퓨팅 최적화 EC2 'c' 인스턴스로 시작하는 것입니다.이 인스턴스 패밀리는 컴퓨팅 집약적인 워크로드에 높은 성능을 제공하기 때문입니다. 또는 게임에서 특정 기능을 구현하는 데 상당한 양의 메모리가 필요한 경우 메모리 최적화 인스턴스가 가장 적합할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

워크로드가 중요한 네트워크 리소스를 활용하는 경우 일반적으로 인스턴스 이름에 'n'을 사용하여 표시되는 네트워크 최적화 인스턴스를 구현하는 것이 좋습니다. 크레딧이 소진되면 성능이 저하되므로 버스트 가능한 인스턴스 유형 't'를 사용하지 마세요. 게임은 지연 시간 및 삭제된 패킷에 민감하므로 EC2 향상된 네트워킹을 사용하여 게임 서버의 네트워크 성능을 개선하는 것이 좋습니다. 향상된 네트워킹은 단일 루트 I/O 가상화(SR-IOV)를 사용하여 지원되는 인스턴스 유형에서 고성능 네트워킹 기능을 제공합니다. SR-IOV는 기존 가상 네트워크 인터페이스에 비해 높은 I/O 성능 및 낮은 CPU 사용률을 제공하는 디바이스 가상화 방법입니다. 향상된 네트워킹을 통해 대역폭과 PPS(Packet Per Second) 성능이 높아지고, 인스턴스 간 지연 시간이 지속적으로 낮아집니다. Elastic Network Adapter를 사용한 향상된 네트워킹은 최신 EC2 인스턴스 유형에 사용할 수 있으며 최신 인스턴스의 성능 향상과 Nitro 하이퍼바이저 개선의 이점을 활용하기 위해 정기적으로 업데이트하는 것이 중요합니다. AWS

게임이 여러 EC2 인스턴스 유형에서 유사하게 작동하는 경우 여러 인스턴스 유형을 사용하여 게임 서버를 호스팅하는 것을 고려해야 합니다. 시간 경과에 따른 성능을 모니터링하고 성능 추세를 식별할 수 있는 충분한 프로덕션 게임 세션을 호스팅한 후 추가 최적화를 수행합니다. 다른 리소스 할당이 필요한 새 기능을 게임에 추가하면 컴퓨팅 요구 사항이 변경될 수 있습니다. 여러 인스턴스 유형을 사용하도록 EC2 Auto Scaling 그룹을 구성하거나, 별도의 Auto Scaling 그룹을 사용하여 별도의 인스턴스 유형을 실행하는 게임 서버 인스턴스를 호스팅하여 지표의 상관 관계 및 집계를 더 쉽게 관리할 수 있습니다.

Intel 기반 인스턴스, AMD 기반 인스턴스, ARM 기반 Graviton 인스턴스와 같은 다양한 유형의 프로세서에서 게임이 어떻게 작동하는지 평가합니다. Unreal Engine 5.1.1 이상은 Graviton용 게임 서버를 컴파일할 수 있으며 게임의 가격 대비 성능을 개선할 수 있습니다. 각 패밀리 내에서 다양한 크기로 스윕 및 포화 테스트를 수행하여 사용률과 성능이 일관된 스위트 스팟을 결정합니다.

또한 컨테이너 및 Lambda 함수를 사용하여 호스팅할 때 게임 성능이 미치는 영향을 벤치마킹해야 합니다. 비동기 게임 및 게임 백엔드 서비스와 같이 수명이 긴 게임 서버 프로세스가 필요하지 않은 사용 사례의 경우 Lambda와 함께 서버리스 아키텍처를 사용하는 것이 좋습니다.이 아키텍처를 사용하면 게임 운영 팀의 관리 및 운영을 간소화하고 게임을 여러에 더 빠르게 배포할 수 있습니다 AWS 리전. 서버리스 모범 사례는 서버리스 애플리케이션 렌즈 - Well-Architected Framework를 참조하세요.

구현 단계

- CPU 집약적 워크로드를 위한 컴퓨팅 최적화 'c' 인스턴스, 메모리 사용량이 많은 작업을 위한 메모리 최적화 인스턴스, 높은 네트워크 처리량을 위한 네트워크 최적화 'n' 인스턴스에서 게임 서버를 벤치마킹합니다.
- 지원되는 인스턴스에서 Elastic Network Adapter(ENA)와 함께 향상된 네트워킹을 사용하여 네트워크 성능을 개선하고, 지연 시간을 줄이고, 패킷 처리 속도를 높일 수 있습니다.
- 여러 인스턴스 유형, 프로세서(Intel, AMD, Graviton), 컨테이너 또는 Lambda 호스팅 옵션을 평가하고 테스트하여 게임 기능이 발전함에 따라 컴퓨팅 솔루션을 조정합니다.

자세한 내용은 글로벌 게임 서버에 적합한 컴퓨팅 전략 선택을 참조하세요.

GAMEPERF05-BP02 non-latency-sensitive 컴퓨팅 작업을 비동기 워크플로로 이동

게임의 성능을 최적화할 때는 클라이언트와 게임 백엔드 간의 일부 상호 작용만 동기식 방식으로 수행해야 한다는 점에 유의해야 합니다. 플레이어 경험의 관점에서 각 기능을 고려하고 특정 상호 작용에 차단 및 리소스 집약적인 동기 통신이 필요한지 또는 이러한 기능을 비동기 방식으로 구현할 수 있는지 결정해야 합니다. 네트워크 호출을 구현할 때는 비동기식 비차단 접근 방식을 사용합니다. 또한 작업을 대기열로 오프로드하고 가능한 경우 클라이언트에 대한 빠른 응답의 우선 순위를 지정하여 효율적인 방식으로 작업을 수행하도록 게임 백엔드를 구성해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

예를 들어, 플레이어 세션이 끝날 때 리더 보드를 업데이트하면 클라이언트가 리더 보드 업데이트가 완료될 때까지 기다릴 필요가 없도록 비동기적으로 구현할 수 있습니다. 대신 게임 클라이언트에서 비동기적으로 이를 구현하고 이러한 유형의 작업을 Amazon SQS와 같은 대기열로 푸시하도록 백엔드 서비스를 설계하는 것이 좋습니다. 이 아키텍처를 사용하면 요청을 수락하도록 백엔드를 구성하고, 비동기 처리를 위해 메시지를 안정적으로 저장하는 데 도움이 되는 SQS에서 대기열에 추가하고, 클라이언

트에 즉시 응답할 수 있습니다. 리더 보드 업데이트가 완료되면 플레이어의 리더 보드 보기가 업데이트되도록 백엔드가 게임 클라이언트에 업데이트를 보낼 수 있습니다.

또는 플레이어가 게임의 리더 보드 화면을 방문하여 최신 데이터를 검색하기만 하면 됩니다. 그러면 백엔드에 웹 요청을 발행하여 캐시에서 최신 데이터를 검색할 수 있습니다.

구현 단계

- 클라이언트-백엔드 상호 작용에 동기 통신이 필요한지 확인하고 가능한 경우 비동기 비차단 접근 방식을 구현하여 리소스 사용을 최적화합니다.
- Amazon SQS를 사용하여 리더보드 업데이트와 같은 중요하지 않은 작업을 오프로드합니다.
- 클라이언트가 온디맨드 또는 백그라운드 업데이트를 통해 최신 리더보드 데이터를 검색하는 등 업데이트된 데이터를 비동기적으로 가져오도록 허용합니다.

리소스

- 마이크로서비스에 대한 비동기식 메시징 이해
- Lambda - 서비스 통합 및 비동기 처리 사용

데이터 관리

GAMEPERF06: 최적의 성능을 위해 게임 서버 로그를 효율적으로 관리 및 분석하고 다양한 유형의 게임 데이터를 저장하려면 어떻게 해야 하나요?

게임에는 가능한 경우 서로 분리해야 하는 플레이어 데이터, 게임 로그 및 서버 로그가 있을 수 있습니다. 로그 수집 및 수명 주기 관리를 중앙 집중화하면 게임 및 서버에서 발생하는 상황에 대한 인사이트를 제공하여 게임 팀에 도움이 될 수 있습니다.

모범 사례

- GAMEPERF06-BP01 로그 수집 및 스토리지 중앙 집중화
- GAMEPERF06-BP02 액세스 패턴에 따라 게임 데이터 분류 및 저장
- GAMEPERF06-BP03 효율적인 로그 형식 지정 및 일괄 처리 활성화
- GAMEPERF06-BP04 로그 교체 및 보존 정책 구현
- GAMEPERF06-BP05 모니터링 및 시각화 도구 사용

GAMEPERF06-BP01 로그 수집 및 스토리지 중앙 집중화

중앙 집중식 로그 수집 및 스토리지 솔루션을 구현하여 게임 서버 인스턴스 및 GameLift에서 로그를 수집합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon CloudWatch Logs와 같은 서비스를 사용하여 게임 서버 및 GameLift 인스턴스에서 로그 데이터를 수집, 모니터링 및 저장합니다. CloudWatch Logs는 로그 관리를 위한 확장 가능하고 완벽하게 관리되는 솔루션을 제공하므로 게임 서버 성능에 영향을 주지 않고 로그 데이터를 효율적으로 저장하고 검색할 수 있습니다. CloudWatch Logs 에이전트를 실행하는 경우 배치 크기, 버퍼 기간과 같은 다양한 설치 유형 및 구성 옵션을 고려하여 게임 서버에 미치는 영향을 최소화합니다. 게임 서버 인스턴스를 임시로 고려하고 가능한 경우 현지화된 로깅에 대한 종속성을 줄입니다. 로깅 모범 사례 구현을 위한 중앙 집중식 정책을 설정합니다.

구현 단계

- Amazon CloudWatch Logs를 사용하여 게임 서버 인스턴스 및 GameLift에서 로그 데이터를 수집, 모니터링 및 저장하여 중앙 집중화되고 확장 가능한 로그 관리를 용이하게 합니다.

GAMEPERF06-BP02 액세스 패턴에 따라 게임 데이터 분류 및 저장

액세스 패턴 및 스토리지 요구 사항에 따라 게임 데이터를 다양한 유형으로 분류합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

일반적인 범주에는 플레이어 데이터, 게임 저장, 영구 월드 스토리지 및 분석 데이터가 포함됩니다.

구현 단계

각 데이터 유형에 적합한 스토리지 솔루션을 사용하여 성능과 비용 효율성을 최적화합니다.

- 플레이어 데이터: 빠르고 확장 가능한 NoSQL 데이터베이스인 Amazon DynamoDB를 사용하여 플레이어 프로필, 기본 설정 및 진행률 데이터를 저장합니다. DynamoDB의 지연 시간이 짧은 액세스 및 자동 조정 기능을 통해 플레이어 데이터를 효율적으로 검색하고 업데이트할 수 있습니다.

- 게임 저장: Amazon S3를 사용하여 게임 저장 및 체크포인트를 저장합니다. S3는 대량의 게임 저장 데이터를 저장할 수 있는 높은 내구성과 확장성을 제공합니다. 게임 저장을 더 빠르게 업로드하고 다운로드하려면 S3 Transfer Acceleration 또는 Amazon CloudFront를 사용하는 것이 좋습니다.
- 영구 월드 스토리지: 영구 월드 상태 또는 공유 게임 데이터가 있는 게임의 경우 Amazon DynamoDB, Amazon ElastiCache 또는 Amazon MemoryDB를 사용하는 것이 좋습니다. ElastiCache 및 MemoryDB는 인 메모리 키-값 저장소를 제공하는 반면 DynamoDB는 SSD 지원 NoSQL 데이터베이스입니다. 이러한 서비스는 저장된 데이터에 대한 빠른 액세스를 제공하여 게임 서버 프로세스가 게임 상태를 저장하는 데 걸리는 시간을 줄여 전체 프로세스 성능을 개선합니다.
- 분석 데이터: Amazon Managed Streaming for Apache Kafka 또는 Kinesis Data Streams를 사용하여 게임 데이터 생산자로부터 데이터 스트림을 수집합니다. Amazon Managed Service for Apache Flink를 실시간 변환 및 분석에 사용하고 Amazon Data Firehose로 전송하여 백엔드 데이터 레이크, 웨어하우스 및 분석 서비스로 처리 및 전송할 수 있습니다. 의 게임 분석 파이프라인 지침은 AWS 서비스가 함께 작동하여 거의 실시간 및 배치 분석을 제공하는 방법을 보여줍니다.

GAMEPERF06-BP03 효율적인 로그 형식 지정 및 일괄 처리 활성화

JSON과 같이 구문 분석할 수 있는 구조화된 형식으로 로그를 생성하도록 게임 서버 프로세스를 구성합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

로그 일괄 처리 기술을 구현하여 게임 서버에서 중앙 집중식 로그 스토리지로 로그 데이터를 전송하는 빈도를 최소화합니다. 로그를 일괄 처리하면 네트워크 오버헤드가 줄어들고 게임 서버 성능이 향상됩니다. 세부 정보 또는 디버그 수준 로그는 가능하면 피해야 할 성능 및 비용 페널티가 발생할 수 있으므로 기본 로그가 아닌 예외로 사용합니다.

GAMEPERF06-BP04 로그 교체 및 보존 정책 구현

로그 데이터의 증가를 관리하고 스토리지 사용률을 최적화하기 위한 로그 교체 및 보존 정책을 설정합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

크기 또는 시간 간격에 따라 로그를 자동으로 교체하도록 게임 서버를 구성합니다. Amazon CloudWatch Logs에서 로그 보존 정책을 정의하여 활성 분석 또는 문제 해결에 더 이상 필요하지 않은 이전 로그 데이터를 자동으로 아카이브하거나 삭제합니다.

GAMEPERF06-BP05 모니터링 및 시각화 도구 사용

모니터링 및 시각화 도구를 사용하여 게임 서버 성능에 대한 인사이트를 얻고 최적화 기회를 식별합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon CloudWatch를 사용하여 주요 지표를 모니터링하고 사전 알림을 위한 경보를 설정합니다. Amazon Managed Service for Prometheus 및 Amazon Managed Grafana와 같은 도구를 활용하여 게임 서버 및 인프라에서 지표를 수집, 쿼리 및 시각화합니다. 유용한 대시보드를 생성하여 성능을 추적하고, 병목 현상을 식별하고, 데이터 기반 최적화를 수행합니다.

네트워킹 및 콘텐츠 전송

GAMEPERF07: 성능을 최적화하기 위해 매치메이킹 서비스를 어떻게 설계하나요?

플레이어 스킬, 인터넷 서비스 제공업체(ISP) 품질 및 플레이어 인구 분포는 성능 튜닝의 차원으로 고려해야 합니다. 게임 세션은 전략적으로 위치한 서버에 배치하여 플레이 필드를 레벨링하고 공정한 게임을 호스팅할 수 있습니다.

모범 사례

- GAMEPERF07-BP01 게임의 네트워크 지연 시간 임계값 정의
- GAMEPERF07-BP02 각 게임 플레이 모드 및 게임 호스팅 리전에 대해 별도의 매치메이킹 서비스 실행
- GAMEPERF07-BP03 매치메이킹 성능 정기 모니터링
- GAMEPERF07-BP04 네트워킹 성능 정기 모니터링
- GAMEPERF07-BP05 네트워크 가속화 기술을 사용하여 인터넷 전반의 성능 향상

GAMEPERF07-BP01 게임의 네트워크 지연 시간 임계값 정의

멀티플레이어 게임을 개발할 때 게임 인프라가 플레이어에게 불필요한 지연 시간을 초래하지 않는지 확인합니다. 게임이 네트워크 지연 시간에 민감한 경우 매치메이킹 로직에서 지연 시간 임계값을 설정하여 가까운 게임 서버 위치에서 호스팅되거나 이상적인 플레이어 경험을 위한 목표를 충족하는 게임 서버 세션에 플레이어를 배치 AWS 리전 하는 우선 순위를 지정해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

많은 지연 시간에 민감한 게임에서는 게임 클라이언트가 각 게임의 인프라 위치를 ping하도록 계측하여 네트워크 지연 시간, 지터 및 패킷 손실과 같은 성능 데이터를 수집하고이 데이터를 지표 수집 백엔드에 보고하여 분석할 수 있도록 하는 것이 일반적입니다. 플레이어를 게임 세션에 매칭할 때 게임 클라이언트의 인식된 네트워크 지연 시간을 게임 서버 인프라에 매치메이킹 서비스 로직에 사용되는 입력 중 하나로 통합하도록 게임을 구성할 수 있습니다.

GAMEPERF07-BP02 각 게임 플레이 모드 및 게임 호스팅 리전에 대해 별도의 매치메이킹 서비스 실행

게임에서 플레이어가 선택할 수 있는 여러 게임 플레이 모드를 제공하는 경우 고유한 요구 사항에 따라 각 게임 플레이 모드의 성능을 독립적으로 조정하고 리소스 경합을 줄일 수 있도록 각 게임 플레이 모드에 대해 매치메이킹 시스템을 분리해야 합니다. 각 게임 플레이 모드에는 허용 가능한 지연 시간, 매치 크기 및 기타 사용자 지정 게임별 매치메이킹 로직에 대한 고유한 요구 사항이 있을 수 있습니다. 또한 다양한 유형의 플레이어를 유치할 수 있습니다. 각 게임 모드의 매치메이킹 서비스를 별도의 소프트웨어 배포로 실행하여 성능을 테스트하고 게임 모드를 독립적으로 운영할 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

예를 들어 이러한 함수를 각 게임 모드에 대해 별도의 Lambda 함수로 실행하거나 별도의 컨테이너 기반 서비스 배포로 운영할 수 있습니다.

게임 서버 위치 근처의 여러 리전에 매치메이킹 서비스를 배포합니다. 플레이어 트래픽에는 많은 경로가 소요되므로 매치메이킹 서비스가 여러 ISPs에서 up-to-date 지연 시간 프로파일을 유지하여 지연 시간이 짧은 게임 세션 배치의 효율성을 개선하는 것이 중요합니다. GameLift FlexMatch는 매치메이커를 위한 리전 선택에 대한 추가 지침을 제공하며, 매치메이커를 다중 리전 게임 세션 대기열과 통합하는 기능을 포함합니다.

GAMEPERF07-BP03 매치메이킹 성능 정기 모니터링

플레이어의 게임 성능을 최적화하는 가장 눈에 띄는 방법 중 하나는 게임 세션에 들어가기 전에 기다려야 하는 시간을 줄이는 것입니다. 대기 시간이 길면 플레이어가 관심을 잃고 이탈할 수 있으므로 매치메이킹 솔루션을 설계할 때 이를 고려하는 것이 중요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임에 대한 매치메이킹 구성을 설계할 때 매치를 구성하는 데 적용되는 조건을 결정하는 규칙을 생성합니다. 이러한 규칙이 시스템 성능, 특히 플레이어의 대기 시간에 미치는 영향을 고려해야 합니다. 새로운 매치메이킹 조건 또는 필터 추가와 같은 변경 사항을 매치메이킹 구현에 배포하기 전에이 변경 사항을 미리 테스트하거나 작은 샘플 플레이어 집단에 카나리 또는 A/B 테스트로 점진적으로 릴리스하여 성능 지표를 수집한 다음 전체 플레이어 집단에이 변경 사항을 도입하는 것이 좋습니다.

각 매치메이킹 요청에 적용된 조건 또는 규칙을 이해하기 위해 세부 로그를 생성하도록 매치메이킹 서비스를 구성합니다. 이렇게 하면 검토에 도움이 되고 필요에 따라 매치메이킹 구현을 조정할 수 있습니다.

예를 들어 Amazon GameLift FlexMatch는 자체 게임 서버 호스팅을 통해 독립 실행형 서비스로 사용하거나 Amazon GameLift에서 호스팅되는 게임 서버와 함께 사용할 수 있는 완전 관리형 매치메이킹 서비스를 제공합니다. FlexMatch는 Amazon EventBridge에 이벤트 알림을 생성할 수 있습니다. FlexMatch 이벤트 알림 설정을 참조하세요. Amazon Simple Notification Service(Amazon SNS)를 사용하여 JSON 형식으로 매치메이킹 데이터를 수신하면 분석을 위해이 정보를 자동으로 처리하고 저장하여 매치메이킹 성능을 개선할 수 있습니다.

매치메이킹 서비스가 플레이어에게 적합한 게임 세션을 찾는 데 걸리는 시간을 추적하는 지표를 설정합니다. 매치메이킹 기간 지표를 정기적으로 검토하고 이러한 시간을 플레이어 행동 및 커뮤니티 감정과 연관시킵니다. 이 데이터를 사용하여 매치메이킹 규칙 구성에 포함될 수 있는 매치메이킹 제한 시간에 적합한 임계값을 개발합니다.

예를 들어 Amazon GameLift FlexMatch는 매치메이킹 요청 제한 시간을 정의하고 시간이 지남에 따라 요구 사항을 완화할 수 있는 매치메이킹 규칙을 생성할 수 있도록 지원합니다. 이 기능을 사용하면 매치메이킹을 생성하여 매치를 쉽게 생성하고 매치를 찾기 어려울 때 플레이어를 게임 세션에 배치할 수 있습니다.

GAMEPERF07-BP04 네트워킹 성능 정기 모니터링

경쟁 게임의 경우 일관된 플레이어 경험을 갖는 것이 중요합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

더 큰 플레이어 기반에 대해 안정적으로 50ms인 게임은 한 플레이어에 10ms ping이 있고 다른 플레이어에 70ms ping이 있는 매치보다 공정하고 재미 있습니다. ISP 라우팅 변경은 플레이어 인구의 일부에 영향을 미칠 수 있으며 매치메이킹 시스템을 조정해야 합니다. Amazon CloudWatch Network Monitoring은 게임 또는 플레이어 인터넷 공급자와 관련된 문제인지 판단하는 데 도움이 됩니다.

구현 단계

- Amazon Cloudwatch Network Monitoring을 사용하여 네트워크 성능을 추적하고 라우팅 문제를 식별합니다.
- VPC 흐름 로그를 사용하여 비정상적인 트래픽 패턴 또는 삭제된 패킷을 식별합니다. 이는 플레이어 지연 시간에 영향을 미치는 네트워크 혼잡, ISP 문제 또는 잘못된 구성을 나타낼 수 있습니다.

GAMEPERF07-BP05 네트워크 가속화 기술을 사용하여 인터넷 전반의 성능 향상

플레이어에게 물리적으로 지연 시간에 민감한 게임 인프라를 배치하는 것 외에도 게임의 네트워크 성능을 최적화하여 플레이어 경험을 개선할 수 있습니다. AWS 는 BGP 프로토콜을 사용하여 인터넷 라우팅에 영향을 미치고 인터넷 서비스 공급자의 경계 네트워크로 향하는 가장 빠른 경로를 사용합니다. 자체 네트워크를 운영하고 라우팅 동작 및 BGP 광고를 더 잘 제어하고 관찰해야 하는 경우 프라이빗 피어링 또는 Direct Connect 를 사용하여 인터넷에서 실행 중인 게임으로 트래픽을 라우팅할 수 있습니다 AWS.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

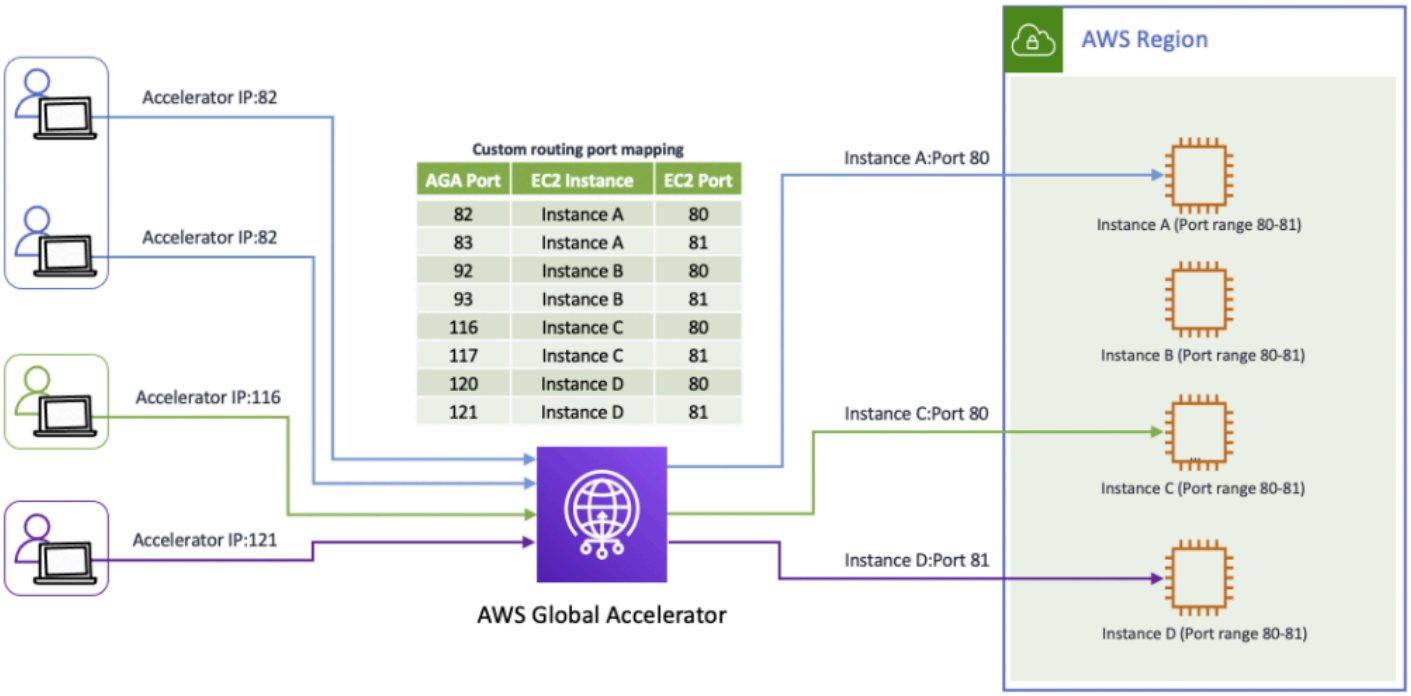
구현 지침

향상된 인터넷 성능과 응답성을 지원하려면 다음 참조 아키텍처를 고려하세요.

Global Accelerator를 사용하여 게임을 위한 네트워크 성능 향상

네트워크 라우팅에 대한 완전관리형 솔루션을 위해 AWS Global Accelerator는 게임 플레이 트래픽, 음성 채팅 및 실시간 메시징 트래픽과 기타 지연 시간에 민감한 애플리케이션을 가속화하는 데 사용할 수 있는 AWS 글로벌 네트워크를 사용하여 애플리케이션의 네트워크 성능을 개선하는 동시에 게임 서버에 빠른 장애 조치를 제공합니다. Global Accelerator 사용자 지정 라우팅 액셀러레이터를 매치메이킹

서비스와 통합하여 정적 애니캐스트 IP 주소 및 포트를 사용하여 여러 플레이어를 동일한 게임 세션으로 결정적으로 라우팅할 수 있습니다.



게임 개발 팀은 전 세계에 배포될 수 있으며 공유 콘텐츠 또는 자산에 대한 성능 있는 액세스가 필요할 수 있습니다. Amazon S3 버킷에 저장된 공유 콘텐츠의 성능을 개선하려면 사용자가 더 가까운 버킷의 데이터에 액세스할 수 있도록 S3 교차 리전 복제를 사용하여 리전 간 데이터의 양방향 복제를 설정할 수 있습니다. 이 액세스 패턴을 간소화하려면 Global Accelerator를 사용하여 글로벌 네트워크를 통해 S3에 대한 요청을 가속화하는 S3 다중 리전 액세스 포인트를 사용합니다. S3

자세한 내용은 AWS Global Accelerator 및 Amazon GameLift FleetIQ를 활용하여 플레이어 경험 개선을 참조하세요.

구현 단계

- AWS Global Accelerator를 사용하면 게임 서버로의 빠른 장애 조치를 촉진하면서 게임 플레이 트래픽, 음성 채팅 및 실시간 메시징의 네트워크 성능을 개선할 수 있습니다.
- 매치메이킹 서비스와 통합되도록 Global Accelerator 사용자 지정 라우팅 액셀러레이터를 구성하여 정적 애니캐스트 IPs를 사용하여 플레이어를 게임 세션으로 결정적으로 라우팅할 수 있습니다.
- S3 리전 간 복제를 활성화하여 분산된 게임 개발 팀을 위해 리전 간에 공유 콘텐츠를 복제합니다.
- S3 다중 리전 액세스 포인트를 사용하여 AWS 글로벌 네트워크를 통해 전 세계에 분산된 사용자의 S3 데이터 액세스를 가속화합니다.

프로세스 및 문화

GAMEPERF08: 플레이어 및 개발자 기대치에 맞게 게임의 성능 기준을 조정하려면 어떻게 해야 하나요?

플레이어와 개발자를 아는 것은 성능 효율성 향상의 가장 중요한 측면 중 하나입니다. 운영 오버헤드가 낮은 성능 있는 게임을 제공하는 것은 플레이어와 개발자에게 경험을 중요하게 생각하고 게임과 스튜디오를 차별화할 수 있음을 보여주는 가장 좋은 방법 중 하나입니다.

모범 사례

- GAMEPERF08-BP01 프로세스에 플레이어를 알리고 포함시킵니다.
- GAMEPERF08-BP02 엔지니어링 팀 기술 및 전문 지식에 맞게 솔루션 선택 조정

GAMEPERF08-BP01 프로세스에 플레이어를 알리고 포함시킵니다.

지연 시간, 초당 프레임 수, 삭제된 패킷과 같은 게임 지표에 표시할 수 있는 옵션을 제공합니다. 상태 페이지와 같은 플레이어 대면 통신을 통해 인프라 문제 및 유지 관리 가동 중지 시간을 해결합니다. 개발 블로그를 포함한 플레이어 커뮤니케이션으로 새로운 게임 위치를 축하하고 예상 플레이어 경험 개선에 대한 기대치를 설정합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

플레이어 포함

관련 파일을 수집하여 게임 클라이언트의 플레이어 지원 티켓에 연결하는 간단한 진단 제출 프로세스를 제공합니다. 플레이어가 서로를 돕고 게임 경험을 개선하는 데 참여할 수 있는 지원 포럼 활성화

장단점과 플레이어 기대치를 고려하세요.

비용 효율성을 위해 백엔드 시스템을 이동하는 것은 플레이어에게 눈에 띄지 않을 수 있지만 게임 서버를 이동하는 것은 ping 시간을 변경할 수 있습니다. 게임 호스팅 위치의 확장 및 축소를 고려하여 플레이어에게 일관되고 공정해야 합니다.

플레이어 커뮤니티와 리전에는 게임의 기대치에 영향을 미칠 수 있는 고유한 특성이 있습니다. 예를 들어 한국은 지구상에서 가장 빠른 인터넷을 보유하고 있으며 게임 플레이에 대한 기대치는 한 자릿수 지

연 시간으로 경쟁이 치열한 플레이를 촉진합니다. 모바일 디바이스에서의 일반적인 게임 플레이는 콘솔 및 PC 세션 플레이와 비교하여 다른 성능 프로파일을 생성하고 패턴을 사용합니다.

로그인 및 로비는 경험의 일부이며 유지 관리를 위해 서버가 오프라인 상태인 경우에도 응답성이 느껴져야 합니다. 로비에서 레이드 나잇을 계획하거나 시간을 보내는 것은 플레이어 경험의 일부이며 성능 효율성을 위해 중점 영역을 선택할 때 고려해야 할 중요한 사항입니다. 플레이어는 몇 달 동안 게임 클라이언트를 열어 둘 수 있으며, 패치 노트를 읽기 위해 가끔 로그인하는 경우도 있습니다. Live Ops 게임은 엔지니어링 프로세스 및 문화의 일환으로 전체 플레이어 경험을 염두에 두어야 합니다.

구현 단계

- 지연 시간, FPS 및 패킷 손실과 같은 게임 내 지표를 제공하고 상태 페이지 및 플레이어 대상 업데이트를 통해 인프라 문제 및 유지 관리 일정을 전달합니다.
- 게임 클라이언트에서 진단 덤프 및 제출 기능을 구현하고 지원 포럼을 생성하여 커뮤니티 기반 문제 해결 및 개선을 촉진합니다.
- 경쟁 리전의 경우 지연 시간이 짧거나 일반 및 장기 세션 플레이어의 경우 응답성이 뛰어난 로그인/로비 환경과 같은 플레이어 커뮤니티 기대치에 맞게 성능을 최적화합니다.
- 활성 게임 플레이에서 유휴 클라이언트 동작에 이르기까지 전체 플레이어 경험을 고려하도록 Live Ops 워크플로를 설계하여 원활한 참여를 촉진합니다.

GAMEPERF08-BP02 엔지니어링 팀 기술 및 전문 지식에 맞게 솔루션 선택 조정

호스팅 옵션을 선택할 때 게임 서버 성능을 관리하고 최적화하는 데 있어 팀의 기술과 전문 지식을 평가합니다. EC2 및 컨테이너와 같은 자체 호스팅 솔루션에는 인프라 관리, 성능 튜닝 및 규모 조정에 대한 더 많은 지식이 필요합니다. 팀에 이러한 기술이 없는 경우 GameLift와 같은 관리형 서비스가 더 적합할 수 있습니다. 많은 복잡성을 추상화하고 팀이 게임별 최적화에 집중할 수 있기 때문입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

이러한 요소를 평가하고 다양한 호스팅 옵션에서 성능 테스트를 수행하면 성능 효율성을 최적화하면서 게임의 특정 요구 사항을 충족하는 가장 적합한 솔루션을 선택할 수 있습니다.

리소스

성능 효율성과 관련된 모범 사례에 대해 자세히 알아봅니다.

관련 문서:

- AWS 아키텍처 센터
- 성능 효율성 원칙 - AWS Well-Architected Framework
- 온프레미스 스토리지 패턴과 AWS 스토리지 서비스 비교
- EC2 인스턴스용 인스턴스 스토어 임시 블록 스토리지
- CloudWatch 에이전트를 사용하여 지표, 로그 및 추적 수집
- CloudWatch 에이전트
- EC2 인스턴스에서 향상된 네트워킹을 켜고 구성하려면 어떻게 해야 하나요?
- AWS Global Accelerator 및 Amazon GameLift FleetIQ를 활용하여 플레이어 경험 개선
- Riot Games 기술 블로그: Valorant의 확장성 및 로드 테스트
- 하이브리드 AWS 솔루션을 사용한 하이퍼 스케일 온라인 게임
- 사용률 포화 및 오류(USE) 메서드
- Amazon EC2 스팟 인스턴스
- Amazon ElastiCache를 사용한 대규모 성능
- Redis를 사용한 데이터베이스 캐싱 전략
- Amazon Virtual Private Cloud 연결 옵션
- 모범 사례 설계 패턴: Amazon S3 성능 최적화

관련 벤치마크:

- Amazon EBS 볼륨 벤치마크
- Amazon EC@ 인스턴스 네트워크 대역폭

관련 도구:

- Unreal Engine: 콘텐츠 테스트 및 최적화
- Unity 프로파일러
- Open 3D Engine(O3DE) 품질 시스템
- Amazon GameLift 서버 모니터링
- Amazon GameLift 테스트 도구 키트

관련 비디오:

- AWS re:Invent 2019: [REPEAT 2] Amazon EC2 파운데이션(CMP211-R2)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- Amazon GameLift FleetIQ 시작하기 - AWS 온라인 테크 토크
- 를 사용하여 게임 개선을 AWS 위한 Zach Blitz의 Riot Games
- AWS re:Invent 2023 - AWS Graviton: AWS 워크로드에 가장 적합한 가격 대비 성능(CMP334)

관련 교육:

- 에서 게임 서버 호스팅 AWS
- 게임 서버에 Amazon GameLiftFleetIQ 사용
- AWS 게임용 시작하기 - 1부
- 에서 게임 서버 호스팅 AWS
- AWS re:Invent 2023 – AWS Graviton: AWS 워크로드에 가장 적합한 가격 성능(CMP334)

비용 최적화

비용 최적화 원칙에는 전체 수명 주기 동안 시스템을 개선하고 개선하는 지속적인 프로세스가 포함됩니다. 이 프로세스는 첫 번째 개념 증명의 초기 설계부터 프로덕션 워크로드의 지속적인 운영에 이르기까지 다양합니다. 이 백서의 사례 개요를 채택하면 최저 가격으로 원하는 비즈니스 성과를 달성하는 비용 인식 시스템을 구축하고 운영할 수 있습니다. 이러한 비용 최적화 관행을 구현하면 기업이 클라우드 투자의 가치를 극대화할 수 있습니다.

게임은 플레이어의 관심과 재생 시간을 위해 경쟁해야 하는 고유한 크리에이티브 프로젝트입니다. 출시 전에는 게임 개발자가 게임이 얼마나 인기 있거나 오래 지속될지 명확하게 이해하지 못하는 경우가 많습니다. 게임의 수익화 전략, 비즈니스 우선순위 및 수명 주기 단계에 따라 개발자는 비용 최적화 결정을 평가할 때 장단점을 만들어야 합니다.

예를 들어, 매우 예상되는 새 게임의 출시 전 단계에서는 일반적으로 speed-to-market, 기능 개발 및 성능에 중점을 둡니다. 우선 순위는 피크 플레이어 수요에 맞게 인프라를 확장할 수 있는지 확인하는 것입니다. 반대로 게임이 성공하지 못하거나 개발 속도가 느려지면 기존 플레이어의 게임을 계속 운영하기 위해 최대한 비용을 절감하는 데 집중할 수 있습니다.

또한 많은 게임 개발자가 여러 게임을 동시에 운영하므로 추가 고려 사항이 필요합니다. 인프라, 소프트웨어 및 직원과 같은 리소스를 여러 라이브 게임에서 공유할 수 있으므로 한 게임의 손실을 다른 게임의 수익으로 상쇄할 수 있습니다. 이 시나리오에서는 비용 최적화에 집중하면 전체 게임 포트폴리오의 재무 상태를 개선할 수 있습니다.

게임의 고유한 비즈니스 모델, 규모 및 예측 불가능성을 고려할 때 다음 주요 질문은 비용 최적화 결정을 안내할 수 있습니다.

- 플레이어, 시스템 및 게임 기능당 인프라 비용을 측정하려면 어떻게 해야 하나요?
- 게임의 현재 수명 주기 단계에서 비용 최적화와 플레이어 경험 간의 적절한 균형은 무엇인가요?
- AWS 리소스에 적합한 요금 모델을 사용하여 투자 수익을 극대화하려면 어떻게 해야 하나요?

이러한 모범 사례를 적용하고 올바른 질문을 하면 게임 개발자가 비용을 최소화하면서 비즈니스 성과를 달성하는 비용 인식 시스템을 구축하고 운영하는 데 도움이 될 수 있습니다.

중점 영역

- 설계 원칙
- 클라우드 재무 관리 시행

- 지출 및 사용량 인식
- 비용 효율적인 리소스
- 데이터 전송 비용
- 수요 및 공급 리소스 관리
- 시간 경과에 따른 최적화
- 리소스

설계 원칙

Well-Architected Framework의 비용 최적화 원칙의 설계 원칙 외에도 다음 설계 원칙은 클라우드에서 게임 워크로드를 실행하는 비용을 최적화합니다.

- 플레이어, 시스템 및 게임 기능당 인프라 비용 측정: 게임 시스템 전반의 특정 플레이어 경험 및 기능에 필요한 인프라 비용을 이해하고 추적합니다. 이를 통해 비용 최적화가 필요할 수 있는 아키텍처 영역을 식별할 수 있습니다.
- 플레이어 경험 대비 비용 최적화의 장단점 평가: 게임이 어떤 단계에 있는지 평가하여 플레이어 경험 또는 비용 최적화라는 올바른 포커스를 결정합니다. 일반적으로 게임이 중요한 질량에 도달하고 플레이어 모집단이 안정화되면 운영 비용 최적화에 집중할 차례입니다. 핵심은 게임 인프라를 가장 비용 효율적인 방식으로 실행할 때 훌륭한 플레이어 경험을 제공하는 것의 균형을 맞추는 것입니다. 이러한 설계 원칙을 구현하면 게임 투자 수익이 극대화됩니다.

클라우드 재무 관리 시행

Games Lens와 관련된 Cloud Financial Management 모범 사례는 없습니다. 클라우드 재무 관리에 대한 지침은 Well-Architected Framework Cost Optimization Pillar를 참조하세요.

지출 및 사용량 인식

GAMECOST01: 게임 환경의 비용을 어떻게 측정하나요?

플레이어당 비용, 게임 기능 및 환경을 이해하면 시간이 지남에 따라 플레이어 수가 변경되고 기능이 추가 및 개선될 때 지출을 관리하고 예측할 수 있습니다. 다양한 게임 환경의 비용을 관리하려면 다음 모범 사례를 고려하세요.

모범 사례

- GAMECOST01-BP01 플레이어, 게임 기능 및 환경당 비용 어트리뷰션 구현
- GAMECOST01-BP02 최적화 기회 발견

GAMECOST01-BP01 플레이어, 게임 기능 및 환경당 비용 어트리뷰션 구현

게임 서버는 일반적으로 인스턴스당 특정 수의 동시 플레이어를 호스팅할 수 있도록 최적화되어 있으므로 게임 서버의 비용 어트리뷰션은 일반적으로 게임 백엔드 서비스보다 성능이 더 간단합니다.이 플레이어는 인스턴스 실행 비용으로 분할 상환할 수 있기 때문입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임 백엔드 서비스의 경우 게임의 구성 요소를 별도의 논리적 또는 물리적 리소스로 관리할 수 있는 고유한 기능으로 분리하여 비용을 쉽게 분석할 수 있도록 하는 것이 좋습니다.

예를 들어, 단일 모놀리식 애플리케이션을 구현하여 게임 백엔드 서비스를 호스팅하는 것이 간단해 보일 수 있지만, 리소스의 컴퓨팅, 네트워킹 및 스토리지 비용이 기능 간에 공유되므로이 패턴으로 인해 더 많은 기능을 추가할 때 시간이 지남에 따라 플레이어 및 게임 기능당 총 비용을 도출하기가 어렵습니다. Amazon Amazon API Gateway Amazon SQS 및 Amazon SNS, 객체 스토리지용 Amazon S3, 데이터베이스 스토리지용 Amazon DynamoDB와 같은 서비스를 사용하여 게임 백엔드 서비스에 서버리스 아키텍처를 채택하는 것이 좋습니다. AWS Lambda AWS Fargate 이러한 서비스는 사용량 기반이며 주로 요청량에 따라 구동되는 요금을 제공하는 제품의 몇 가지 예에 불과하므로 비용을 세부적으로 시각화할 수 있습니다. Lambda 함수, Fargate 서비스, DynamoDB 테이블 및 S3 버킷과 같은 개별 리소스를 비용 할당 태그와 연결할 수 있으므로 각 서비스의 비용을 쉽게 이해할 수 있는 게임 기능 이름으로 이러한 서비스의 비용을 귀속할 수 있습니다.

또한 다양한 환경에 대한 비용을 귀속할 수 있도록 각 게임 개발 환경을 별도로 관리하는 것이 좋습니다. 일반적으로 게임 개발자는이 게임 산업 렌즈의 운영 원칙에 설명된 대로 개발, 테스트, 스테이징 및 프로덕션 환경을 위한 별도의 환경을 관리합니다. 각 환경은 일반적으로 확장성, 성능 및 사용량 요구 사항이 다르며 별도의 팀에서 관리할 수 있습니다. 비용을 제어하려면 각 환경의 비용을 적절하게 모니터링하고 속성을 지정할 수 있도록 이러한 환경을 구성합니다.

자세한 내용은 다음 설명서를 참조하세요.

- 확장 가능한 서버리스 멀티플레이어 게임 빌드
- WebSockets 기반 백엔드가 있는 독립 실행형 게임 세션 서버

- 서버리스 백엔드가 있는 독립 실행형 게임 세션 서버

구현 단계

- AWS Lambda Amazon API Gateway 및와 같은 서버리스 또는 컨테이너화된 아키텍처를 사용하여 게임 백엔드 서비스를 고유한 기능으로 분리 AWS Fargate 하여 기능당 세분화된 비용 어트리뷰션을 활성화합니다.
- 비용 할당 태그를 개별 리소스(예: Lambda 함수, DynamoDB 테이블 및 S3 버킷)에 적용하여 비용을 특정 게임 기능과 연결하여 비용 분석을 개선합니다.
- 개발, 테스트, 스테이징 및 프로덕션을 위한 별도의 환경을 관리하고 확장성 및 사용 요구 사항에 맞게 비용을 독립적으로 구성하고 모니터링합니다.

GAMECOST01-BP02 최적화 기회 발견

게임 개발자와 게시자는 AWS FinOps 관행을 사용하여 클라우드 비용을 최적화하고 클라우드 지출에 대한 가시성을 높일 수 있습니다. 이렇게 하면 게임 생산자는 플레이어의 인프라를 유지하는 데 필요한 평균 비용을 게임에서 제공하는 재무 결과에 맞출 수 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

AWS 는 클라우드 서비스 비용을 관리하고 최적화하기 위해 Cloud Financial Management에 사용할 수 있는 솔루션 지침을 제공합니다. 이 기능에는 지출 대시보드, 최적화, 지출 한도, 비용 청구, 이상 탐지 및 대응과 같은 주제에 대한 의사 결정을 지원하는 세분화된 가시성과 비용 및 사용량 분석이 포함됩니다. Cloud Financial Management의 솔루션 지침에는 예산 및 예측 기능이 포함되어 있어 워크로드에 대해 정의된 비용 최적화 아키텍처를 제공하므로 팀과 관련된 올바른 요금 모델 및 속성 리소스 비용을 선택할 수 있습니다. 이렇게 하면 환경 및 리소스 전반에서 추적, 알림 및 비용 최적화 기술이 활성화됩니다. 비용 정보를 중앙에서 관리하고 중요한 이해관계자에게 대상 가시성을 확보하고 의사 결정을 지원하는 데 필요한 액세스 권한을 부여할 수 있습니다.

또 다른 주요 FinOps 도구는 Cost Optimization Hub입니다. Cost Optimization Hub는 및에서 비용 최적화 권장 사항 및 기회를 중앙에서 볼 수 AWS 계정 AWS 리전있으므로 AWS 지출을 최대한 활용할 수 있습니다. Cost Optimization Hub를 사용하여 AWS 계정 및에서 AWS 비용 최적화 권장 사항을 식별, 필터링 및 집계할 수 있습니다 AWS 리전. Cost Optimization Hub는 리소스 적정 크기 조정, 유휴 리소스 삭제, 절감형 플랜, 예약 인스턴스에 대한 권장 사항을 제공합니다. 단일 대시보드를 사용하면 여러 AWS 제품으로 이동하여 비용 최적화 기회를 식별할 필요가 없습니다.

게임 팀이 AWS Management Console 홈의 myApplications 공유 AWS 계정 를 사용하는 경우를 사용하여 개별 워크로드에 대한 애플리케이션 리소스 비용을 볼 수 있습니다. 이 세분화된 보기를 통해 게임 인프라 내의 특정 비용 추세를 식별할 수 있으므로 리소스 할당 및 최적화에 대해 정보에 입각한 결정을 내릴 수 있습니다.

또한 AWS Data Exports를 사용하여 결제 및 비용 관리 데이터를 정기적으로 검토하면 숨겨진 비용 절감 기회가 발견됩니다. 이 세부 보고서는 클라우드 지출을 포괄적으로 분석하여 리소스가 과다 지출되고 활용되지 않는 영역과 보다 비용 효율적인 서비스 또는 요금 모델을 활용할 수 있는 기회를 식별할 수 있습니다.

AWS게임 개발자와 게시자는 FinOps 원칙을 수용하고에서 제공하는 도구를 활용하여 클라우드 리소스를 가장 효율적으로 활용하여 궁극적으로 수익을 높이고 추가 게임 개발 및 혁신을 위한 자금을 확보할 수 있습니다.

구현 단계

- AWS 클라우드 재무 관리 도구를 사용하여 세분화되고 세부적인 가시성, 지출 대시보드, 이상 탐지 및 비용 어트리뷰션을 통해 클라우드 비용을 효과적으로 최적화하고 추적할 수 있습니다.
- Cost Optimization Hub를 사용하여 AWS 계정 및 리전에서 규모 조정, Savings Plans 및 예약 인스턴스 권장 사항을 중앙 집중화할 수 있습니다.
- 의 Data Exports 및 MyApplication AWS 을 사용하여 AWS 결제 데이터를 정기적으로 검토하여 워크로드별 비용을 분석하고, 절감 기회를 발견하고, 리소스 할당을 최적화할 수 있습니다.

비용 효율적인 리소스

GAMECOST02: 게임 서버에 적합한 컴퓨팅 솔루션을 어떻게 선택하나요?

다른 유형의 워크로드에 비해 게임 워크로드의 가장 고유한 측면 중 하나는 게임 서버입니다. 플레이어가 게임 클라이언트에서 게임 세션을 플레이하기 위해 게임 서버에 연결되므로 게임 서버는 플레이어 경험에 매우 중요합니다.

또한 게임 서버는 멀티플레이어 게임을 운영하기 위한 가장 큰 비용 동인 중 하나입니다. 따라서 게임 서버의 컴퓨팅 인프라를 활용하여 비용을 절감하는 방법을 최적화하는 것이 중요합니다.

모범 사례

- GAMECOST02-BP01 인터넷을 통한 데이터 전송 비용 최적화

- GAMECOST02-BP02 각 게임 서버 인스턴스에서 호스팅되는 게임 세션 수를 최적화하여 비용 최적화
- GAMECOST02-BP03 적절한 컴퓨팅 요금 옵션을 선택하여 비용 절감

GAMECOST02-BP01 인터넷을 통한 데이터 전송 비용 최적화

AWS 주로 AWS 리소스에서 인터넷으로의 아웃바운드(송신) 데이터 전송에 대해 요금이 부과되지만 게임 회사는 AWS Direct Connect 또는 AWS 게이트웨이 로드 밸런서를 통한 데이터 전송과 관련하여 높은 비용을 부담할 수 있으며, 이는 인바운드(수신) 및 아웃바운드 데이터 모두에 대해 요금이 부과될 수 있습니다. 게임 AWS 백엔드에서 플레이어로 데이터를 전송하는 데 드는 전체 비용을 줄이는 솔루션을 구현합니다.이 솔루션은 AWS 리소스에서 송신 요금을 최소화하고 AWS 연결 서비스를 통해 수신 및 송신 요금을 관리하는 옵션을 평가하는 데 중점을 둡니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

Amazon CloudFront를 사용하여 콘텐츠 전송 및 퍼블릭 웹 애플리케이션 비용을 절감할 수 있습니다.

클라우드에 저장된 게임 콘텐츠 및 자산은 일반적으로 Amazon S3에 저장되며 S3에서 직접 또는 Amazon S3에서 콘텐츠를 검색Amazon S3하여 클라이언트에 전달하는 Amazon EC2에서 호스팅되는 웹 서버에서 게임 클라이언트로 전달됩니다. 콘텐츠 다운로드의 데이터 전송 비용을 줄이려면 클라우드 스토리지 앞에 Amazon CloudFront를 사용하여 사용자에게 콘텐츠를 제공하는 것이 좋습니다.

CloudFront를 사용하면 CloudFrontpoints-of-presence 콘텐츠를 리전에서 직접 전송하는 것보다 비용이 적게 들기 때문에 데이터 전송 비용을 줄일 수 있으며, CloudFront는 Amazon EC2 및 Amazon S3와 같은 AWS기반 오리진에 대한 오리진 검색 요금을 부과하지 않습니다. 콘텐츠가 정적이고 자주 변경되지 않는 경우 CloudFront를 사용하여 해당 데이터를 최종 사용자에게 더 가깝게 캐싱하여 비용을 더욱 절감할 수 있습니다.

또한 CloudFront는 AWS 네트워크를 통해 트래픽을 라우팅하여 서버와 클라이언트 간의 데이터 전송 비용을 줄일 수 있으므로 캐싱을 사용하지 않더라도 프런트페이스 퍼블릭 웹 애플리케이션 및 서비스의 비용 효율성을 개선합니다.

Amazon CloudWatch를 사용하여 Amazon CloudFront 사용량을 모니터링할 수 있습니다. 여러 콘텐츠 전송 네트워크(CDNs)를 사용하는 사용 사례의 경우 Amazon CloudFront Origin Shield는 추가 캐싱 계층을 제공하여 서로 다른 공급자의 오리진 요청 수를 통합하고 줄일 수 있습니다.

게임 네트워크 트래픽을 이해하기 위해 VPC 흐름 로그 및 Amazon CloudWatch Internet Monitor를 활성화하여 플레이어 또는 게임 백엔드 연결에 대한 end-to-end 엔드 가시성을 확보할 수 있습니다. 이 접

근 방식은 데이터 전송 비용이 높은 원인을 식별하고 아키텍처 변경을 수행하여 데이터 전송 지출을 최적화할 수 있습니다.

구현 단계

- Amazon S3 또는 EC2-based 콘텐츠 오리진 앞에서 Amazon CloudFront를 사용하여 CloudFront points-of-presence 더 저렴한 비용으로 전송하고 오리진 검색 요금을 제거하여 데이터 전송 비용을 절감할 수 있습니다.
- VPC 흐름 로그 및 Amazon CloudWatch Internet Monitor를 활성화하여 네트워크 트래픽을 분석하고 아키텍처 변경을 식별하여 데이터 전송 비용을 최적화합니다.
- CloudFront Origin Shield를 구현하여 추가 비용 효율성을 위해 여러 CDNs 요청을 통합하고 줄입니다.

콘텐츠 전송에 대한 자세한 모범 사례는 게임용 콘텐츠 전송 백서를 참조하세요.

GAMECOST02-BP02 각 게임 서버 인스턴스에서 호스팅되는 게임 세션 수를 최적화하여 비용 최적화

서버 인스턴스당 호스팅되는 게임 세션 수를 최적화하여 컴퓨팅 사용률을 높이고 컴퓨팅 인프라 비용을 절감합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

비용을 최적화하기 위해 게임 개발자는 게임 서버의 패킹 밀도라고도 하는 동일한 물리적 또는 가상 서버에서 호스팅되는 게임 세션 수를 극대화해야 합니다. 이는 인스턴스에서 동시에 호스팅할 수 있는 게임 서버 프로세스의 수를 늘려 달성됩니다.

단일 게임 서버 프로세스에서는 일반적으로 EC2 인스턴스에서 사용 가능한 전체 리소스를 사용할 필요가 없습니다. 이는 게임의 컴퓨팅 비용을 줄이는 가장 중요한 방법 중 하나이며 별도의 포트에서 EC2 인스턴스에서 여러 서버 프로세스를 생성하고 관리할 수 있는 소프트웨어를 사용해야 합니다.

예를 들어 Amazon GameLift에는 인스턴스당 최대 게임 서버 프로세스 수에 대한 할당량이 있으므로 호스팅 비용을 줄일 수 있도록 활용해야 합니다. 자세한 내용은 Amazon GameLift Servers 엔드포인트 및 할당량에서 인스턴스당 최대 게임 서버 프로세스의 현재 할당량에 대한 세부 정보를 참조하세요.

EC2 인스턴스와 같은 가상 머신에 게임 서버 프로세스를 배포하는 대신 게임 개발자는 컨테이너 오케스트레이션 솔루션을 사용하여 게임 서버를 컨테이너 기반 애플리케이션으로 실행하는 것이 인기를

얻고 있습니다. 게임 개발자는 Amazon EKS에서 Amazon Elastic Container Service(Amazon ECS) 또는 아곤 및 오픈 매치를 사용한 게임 서버 호스팅 지침을 사용할 수 있습니다. 또 다른 옵션은 ECS와 EKS 모두에서 작동하는 서버리스 컴퓨팅 엔진인 Game Server Hosting on AWS Fargate입니다. 이를 통해 기본 인프라를 관리할 필요 없이 게임에 집중할 수 있습니다.

컨테이너 솔루션은 지정한 리소스 요구 사항 및 기타 배치 로직에 따라 게임 서버 컨테이너를 호스팅하기 위해 클러스터에서 사용 가능한 컨테이너 인스턴스를 자동으로 찾을 수 있는 작업 예약 기능을 제공합니다. 그러나 활성 플레이어 세션을 방해하지 않는 방식으로 조정 및 플레이어 배치 동작을 관리하는 방법을 고려하는 것이 중요합니다.

구현 단계

- 별도의 포트와 프로세스 관리 소프트웨어를 사용하여 EC2 인스턴스당 여러 게임 서버 프로세스를 실행하여 패킹 밀도를 높입니다.
- Amazon GameLift 또는 ECS, EKS 또는 같은 컨테이너 솔루션을 사용하여 게임 서버 프로세스를 효율적으로 관리하고 인프라 비용을 절감 AWS Fargate 할 수 있습니다.
- 리소스 사용률을 지속적으로 모니터링하여 플레이어 경험을 손상시키지 않으면서 패킹 밀도를 개선하고 비용 효율성을 유지합니다.

GAMECOST02-BP03 적절한 컴퓨팅 요금 옵션을 선택하여 비용 절감

다양한 인스턴스 유형 및 컴퓨팅 옵션에서 게임 서버 소프트웨어의 성능 테스트를 실행하여 게임에 가장 비용 효율적인 옵션을 결정합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

워크로드에 적합한 EC2 인스턴스 유형을 효율적으로 활용하는 것 외에도 비용 최적화 목표에 가장 적합한 컴퓨팅 요금 옵션을 고려하세요. 온디맨드 인스턴스, 스팟 인스턴스, 예약 인스턴스, Savings Plans.

Savings Plans(SPs)은 사용량 약정을 통해 컴퓨팅에 대한 할인을 제공하며 1년 또는 3년 기간 동안 예상 사용량을 예측할 수 없는 시나리오에 적합합니다. 예약 인스턴스와 같은 할인을 제공하며 리전, 인스턴스 패밀리, 운영 체제, 테넌시 간에 이러한 할인을 유연하게 적용할 수 있습니다. 또한 일반 게임의 게임 서버 호스팅 옵션이 될 수 AWS Fargate있는 사용자 또는 게임 서버가 필요하지 않은 턴 기반 게임의 훌륭한 옵션으로 사용되는 AWS Lambda 사용자에게도 적용할 수 있습니다. 자세한 내용은 확장되는 서버리스 멀티플레이어 게임 구축을 참조하세요.

Savings Plans은 게임이 출시될 때 EC2 인스턴스에 기여하는 게임 서버 워크로드에 대한 비용을 절감하기 위해 게임 출시 중에 도입됩니다. Savings Plans은 게임 운영 팀이 게임이 장기간 프로덕션 환경에서 실행된 후 플레이어 트래픽을 더 잘 이해할 수 있을 때 출시 후 도입될 수도 있습니다.

Savings Plans은 리전별 유연성을 제공하므로 여러 리전에서 예측할 수 없는 사용량으로 게임에 대한 게임 서버 지출을 최적화하는 데 특히 적합합니다.

예를 들어 일일 플레이어 사용 패턴에 플레이어 기반을 지원하기 위해 최소 20개의 서버가 필요하지만 주기적으로 최대 40개의 서버가 필요한 경우 절감Savings Plan 약정을 구매하여 20개의 서버의 기준을 충족하는 것이 좋습니다. 해당 사용량 수요는 예측 가능하고 일관되며 구매한 사용량 약정을 최대한 활용할 수 있기 때문입니다.

Savings Plans의 사용률을 극대화하고 온디맨드 및 스팟 인스턴스와 같이 예측할 수 없는 게임 서버 사용량 급증에 대한 유연성을 제공하는 다른 구매 옵션으로 절감형 플랜을 강화하여 최적의 절감 효과를 달성합니다.

스팟 인스턴스는 가장 큰 컴퓨팅 할인을 제공하고, 사용량 약정이 필요하지 않으며, 예측할 수 없고 급증하는 워크로드 유형에 대한 유연성을 제공하므로 게임 서버를 실행하는 데 적합합니다. 그러나 스팟 인스턴스는 중단될 수 있으므로 게임 세션 기간이 짧거나 중단에 대한 허용 오차가 더 높은 게임 서버 워크로드에 가장 적합합니다.

EC2 스팟 인스턴스와 함께 Amazon EKS에서 Kubernetes를 사용하여 게임 서버를 실행하는 방법에 대한 자세한 내용은 Aurora Serverless를 사용하여 EC2 스팟으로 대규모 멀티플레이어 게임을 실행하는 방법을 참조하세요.

Amazon EC2 스팟 인스턴스를 사용하여 온디맨드 요금에 비해 최대 절감 효과를 제공하는 중단 가능성이 가장 낮은 풀을 결정합니다.

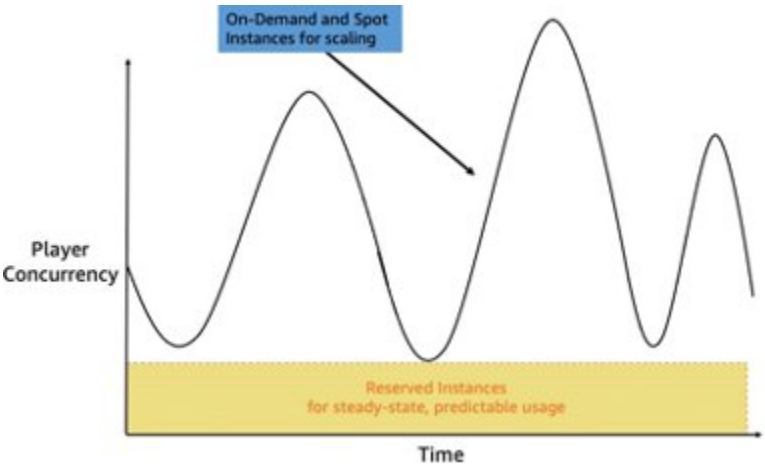
스팟을 사용할 때는의 여러 EC2 인스턴스 유형 및 가용 영역에서 게임 서버 워크로드를 실행 AWS 리전 하여 용량 사용량을 다각화하고 중단 위험을 줄이는 것이 좋습니다.

스팟 인스턴스를 온디맨드 인스턴스와 함께 사용하여 활성 게임 세션에 대한 잠재적 중단의 영향을 최소화하고 용량 최적화 할당 전략을 사용하여 중단 위험을 더욱 줄이는 것이 좋습니다.

추가 모범 사례는 Amazon EC2 스팟 모범 사례를 참조하세요. 위험한 스팟 인스턴스를 대체하기 위한 Auto Scaling의 용량 리밸런싱을 사용하여 스팟 인스턴스의 중단 위험이 증가할 때 사전 예방적으로 용량을 모니터링하고 추가할 수 있습니다.

Amazon GameLift FleetIQ는 스팟 인스턴스와 통합되어 중단 위험을 줄이면서 저렴한 스팟 인스턴스 사용을 최적화합니다. GameLift를 사용하여 게임을 호스팅하는 경우 컴퓨팅 리소스 선택에 대한 GameLift 설명서를 검토하세요. 자세한 내용은 관리형 플릿의 컴퓨팅 리소스 선택을 참조하세요.

다음 다이어그램은 게임 서버 워크로드에 여러 컴퓨팅 요금 옵션을 사용하는 방법을 보여주는 예제를 제공합니다.



여러 EC2 요금 옵션으로 게임 서버 호스팅

다이어그램에서 플레이어 동시성은 시간이 지남에 따라 변동하므로 사용률을 관리하고 비용 최적화를 달성하기가 어렵습니다. 이러한 변동을 해결하려면 EC2용 Savings Plans을 사용하여 최소 사용 요구 사항을 충족하는 동시에 플레이어 요구 사항을 충족하기 위해 EC2 온디맨드 및 EC2 스팟 인스턴스를 사용하는 등 다양한 컴퓨팅 요금 옵션을 혼합하여 채택하는 것이 좋습니다.

구현 단계

- 예측 가능한 기준 사용량에 Savings Plans 사용하고 사용량 급증 시 유연성과 비용 최적화를 위해 스팟 및 온디맨드 인스턴스와 결합합니다.
- 세션 지속 시간이 짧거나 중단 허용 오차가 높은 게임 서버에 스팟 인스턴스를 사용하여 인스턴스 유형과 가용 영역을 다각화하여 위험을 최소화합니다.
- EC2 스팟 인스턴스 어드바이저, 용량 리밸런싱, GameLift FleetIQ와 같은 도구를 구현하여 스팟 인스턴스 사용을 최적화하고 중단을 사전에 관리합니다.

데이터 전송 비용

GAMECOST03: 게임 인프라의 데이터 전송 비용을 어떻게 최적화하고 있나요?

게임은 플레이어의 게임 클라이언트 디바이스와 게임 인프라 간에, 그리고 게임 인프라의 구성 요소 간에 상당한 양의 데이터를 인터넷을 통해 전송할 수 있습니다.

예를 들어, 데이터 전송은 플레이어가 게임 콘텐츠 업데이트를 게임 클라이언트에 다운로드하고, 게임 진행 상황을 클라우드에 저장하고, 친구와 실시간 멀티플레이어 게임 세션에 참여하고, 게임 인프라가 리전과 가용 영역 간에 데이터를 전송할 때 발생합니다. 아키텍처 선택을 최적화하여이 데이터 전송 비용을 줄이려면 게임 워크로드에서 데이터 전송이 발생하는 위치를 이해하는 것이 중요합니다.

게임 워크로드의 데이터 전송 비용을 최적화하려면 다음 모범 사례를 고려하세요.

모범 사례

- GAMECOST03-BP01 사용자가 생성한 콘텐츠에 적합한 스토리지 유형을 선택하여 비용 절감
- GAMECOST03-BP02 게임 백엔드에 데이터베이스 최적화

GAMECOST03-BP01 사용자가 생성한 콘텐츠에 적합한 스토리지 유형을 선택하여 비용 절감

게임에서 생성 및 저장된 각 데이터 유형에는 워크로드에 적합한 스토리지 솔루션을 결정할 때 고려해야 할 고유한 특성이 있습니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 낮음

구현 지침

Amazon S3 객체 수명 주기 관리를 사용하여 가장 비용 효율적인 스토리지 클래스에 객체 데이터를 저장합니다. Amazon S3는 여러 스토리지 클래스와 객체 수명 주기 관리를 제공하므로 간단하고 세분화된 정책을 설정하여 스토리지 계층 간에 데이터를 자동으로 전환하여 비용을 절감할 수 있습니다. 기본적으로 S3 표준 스토리지 클래스에 데이터를 저장하는 대신 수명 주기 구성을 설정하여 시간이 지남에 따라 계층 간에 데이터를 자동으로 전환하거나, 알 수 없거나 변화하는 액세스 패턴에 S3 Intelligent-Tiering 스토리지 클래스를 사용하는 것이 좋습니다.

또는 S3 Intelligent-Tiering은 수명 주기 정책을 수동으로 설정할 필요 없이 비용 최적화를 제공하고 이제는 작고 수명이 짧은 객체에 가장 적합하므로 계층 간에 데이터를 비용 효율적이고 자동으로 전환할 수 있으며 기본 스토리지 클래스로 권장됩니다. 자세한 내용은 Amazon S3 Intelligent-Tiering – Short-Lived 및 Small Objects의 향상된 비용 최적화를 참조하세요.

Amazon S3의 일반적인 사용 사례에는 게임 자산, 정적 콘텐츠, 게임 로그, 데이터 레이크 스토리지 및 백업의 스토리지가 포함됩니다. 개발 중에 공유 파일 시스템을 워크스테이션에 연결하는 등 파일 시스템이 필요한 사용 사례의 경우 인프라를 관리할 필요 없이 파일을 추가하고 제거할 때 다양한 스토리지 클래스를 제공하고 자동으로 확장 및 축소되는 Amazon Elastic File System(Amazon EFS)을 사용하는 것이 좋습니다.

Amazon S3 One Zone-IA는 게임 내 세션, 매치메이킹 또는 필요에 따라 다시 생성할 수 있는 기타 임시 정보와 관련된 임시 데이터에 이상적인 스토리지 옵션입니다. 이러한 유형의 게임 데이터는 여러 가용 영역(AZs) 간의 중복성이 필요하지 않습니다. 이 저렴한 스토리지 클래스는 분석 또는 디버깅에 사용되는 플레이어 작업, 게임 이벤트 및 기타 원격 측정 데이터의 레코드에 적합합니다.

이러한 게임 데이터에 S3 Express One Zone을 사용할 때 얻을 수 있는 주요 비용 최적화 이점은 표준 S3 스토리지 클래스에 비해 스토리지 비용이 최대 20% 절감된다는 것입니다. 이는 미션 크리티컬 애플리케이션 데이터와 동일한 수준의 내구성과 가용성이 필요하지 않은 대량의 데이터가 있는 게임에 특히 유리할 수 있습니다. 게임 개발자와 게시자는 S3 One Zone을 활용하여 전반적인 플레이어 경험을 손상시키지 않고 클라우드 스토리지 비용을 최적화할 수 있습니다.

구현 단계

- 스토리지 클래스 간에 데이터를 전환하거나 액세스 패턴이 변화하는 자동 비용 최적화를 위한 기본값으로 S3 Intelligent-Tiering을 사용하도록 Amazon S3 수명 주기 정책을 구성합니다.
- 원격 측정 및 매치메이킹 레코드와 같은 임시 게임 세션 데이터에 S3 One Zone-Infrequent Access를 사용하면 충분한 가용성을 유지하면서 스토리지 비용을 최대 20% 절감할 수 있습니다.
- 개발 중에 공유 파일 시스템이 필요한 경우 Amazon EFS를 사용하여 탄력적 용량과 여러 스토리지 클래스로 스토리지 관리를 간소화합니다.

GAMECOST03-BP02 게임 백엔드에 데이터베이스 최적화

게임은 데이터베이스에 크게 의존하여 플레이어 프로필 및 인벤토리부터 게임 내 마이크로 트랜잭션 및 진행 지표에 이르기까지 다양한 중요 데이터를 저장합니다. 또한 데이터베이스는 플레이어 그룹, 파티를 생성 및 유지 관리하고 조절 정책을 적용하는 등 게임의 사회적 측면을 관리하는 데 중요한 역할을 합니다. 게임의 플레이어 기반이 증가함에 따라 증가하는 데이터 및 사용량 수요를 수용하기 위해 관련 데이터베이스 비용이 필연적으로 증가할 것입니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 중간

구현 지침

Amazon Aurora에서 실행되는 게임 백엔드의 경우 몇 가지 비용 최적화 전략을 사용할 수 있습니다. 한 가지 주요 권장 사항은 사용 패턴에 따라 읽기 전용 복제본을 자동으로 조정하여 트래픽 변동을 처리하기 위해 복제본 수를 동적으로 늘리거나 줄이는 것입니다. 즉, 실제로 필요한 리소스에 대한 비용을 지불하게 됩니다. 또 다른 최적화 전략은 게임 분석에 사용되는 읽기 전용 복제본을 Amazon S3로 DB 스냅샷 내보내기로 대체하는 것입니다. S3 스토리지 서비스는 일반적으로 프로비저닝된 Aurora 데이터

베이스 인스턴스보다 저렴하기 때문입니다. 자세한 내용은 Amazon Amazon S3 RDS로 DB 스냅샷 데이터 내보내기를 참조하세요.

코어 데이터베이스 인스턴스에 Amazon Aurora용 예약 DB 인스턴스를 사용하고 Aurora Serverless 구성으로 전환하면 리소스 사용률을 보다 유연하고 세밀하게 제어하여 상당한 장기 비용 절감 효과를 얻을 수도 있습니다.

마찬가지로 Amazon DynamoDB를 사용하는 게임 백엔드의 경우 DynamoDB 온디맨드 용량 모드를 사용하는 것이 특히 신규 또는 예측할 수 없는 워크로드의 경우 효과적인 선택이 될 수 있습니다. 과도하게 프로비저닝할 필요 없이 소비하는 리소스에 대해서만 비용을 지불할 수 있기 때문입니다. 시간이 지남에 따라 게임 트래픽 패턴이 더 안정적이고 예측 가능하게 되면 DynamoDB 프로비저닝된 용량 모드로 전환하여 더 나은 용량 계획을 통해 비용을 절감할 수 있습니다. DynamoDB 테이블에서 Auto Scaling을 활성화하면 트래픽 변동에 따라 서비스가 프로비저닝된 용량을 동적으로 조정할 수 있는 또 다른 키 최적화입니다. 시작하기 전에 개발 환경에서 게임의 데이터 구조를 테스트하여 불필요한 로컬 보조 인덱스(LSIs)와 글로벌 보조 인덱스(GSIs 찾아 제거합니다. 이로 인해 게임 데이터 스토리지 및 운영 비용이 크게 절감될 수 있습니다. 더 많은 대상 쿼리를 위해 게임 백엔드 코드에서 비효율적인 스캔 작업을 제거하고, Amazon DynamoDB 예약 용량을 구매하고, AWS Lambda 트리거가 있는 DynamoDB 스트림을 활용하여 게임 백엔드 이벤트를 처리하면 DynamoDB 비용을 더욱 최적화할 수 있습니다. 자세한 내용은 DynamoDB의 데이터 쿼리 및 스캔 모범 사례를 참조하세요.

Amazon Aurora와 DynamoDB 모두에 이러한 비용 최적화 전략을 구현하면 게임 개발자와 게시자가 게임 백엔드 데이터베이스 비용을 크게 줄일 수 있습니다.

구현 단계

- 변동하는 트래픽 및 분석 요구 사항을 비용 효율적으로 처리하려면 Amazon S3로 Aurora 읽기 전용 복제본 자동 크기 조정 및 DB 스냅샷 내보내기를 사용합니다.
- 새 워크로드의 온디맨드 용량으로 시작하고, 예측 가능한 트래픽을 위한 Auto Scaling을 통해 프로비저닝된 용량으로 전환하고, 미사용 LSIs 및 GSI를 제거하여 DynamoDB 비용을 최적화합니다. GSIs
- 대상 쿼리에 유리한 비효율적인 스캔 작업을 피하고, 예약 인스턴스 또는 예약 용량을 사용하고, 이벤트 처리를 AWS Lambda 위해 DynamoDB Streams를와 함께 사용합니다.

수요 및 공급 리소스 관리

Games Lens와 관련된 수요 및 공급 리소스 관리 모범 사례는 없습니다.

수요 관리 및 리소스 공급에 대한 자세한 내용은 Cost Optimization Pillar - AWS Well-Architected Framework를 참조하세요.

시간 경과에 따른 최적화

Games Lens와 관련된 시간 경과에 따른 최적화 모범 사례는 없습니다.

시간 경과에 따른 비용 최적화에 대한 자세한 내용은 Cost Optimization Pillar - AWS Well-Architected Framework를 참조하세요.

리소스

비용 최적화 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

관련 문서:

- Amazon VPC에서 NAT 게이트웨이에 대한 데이터 전송 요금을 줄이려면 어떻게 해야 합니까?
- Agones용 Amazon GameLift FleetIQ 어댑터 소개
- Amazon VPC에서 NAT 게이트웨이 트래픽에 대한 상위 기여자를 찾으려면 어떻게 해야 합니까?
- 글로벌 게임 서버에 적합한 컴퓨팅 전략 선택
- AWS Well-Architected Labs - 비용 효율적인 리소스
- Amazon VPC CNI 플러그인은 노드 제한당 포드를 증가시킵니다.
- 비용 최적화를 위한 아키텍처 모범 사례
- Amazon SageMaker AI RL 및 Amazon EKS를 사용하여 플레이어 대기 시간 단축 및 적절한 컴퓨팅 할당 크기 조정
- AWS Compute Optimizer
- Electronic Arts, Amazon S3 Intelligent-Tiering 및 Amazon Glacier를 사용하여 스토리지 비용 및 운영 최적화
- Windows 워크로드를 Linux로 마이그레이션하여 친숙하지 않은 라이선스 사례 이스케이프
- Overview of Data Transfer Costs for Common Architectures
- AWS 및 Kubecost가 협력하여 EKS 고객에게 비용 모니터링 제공
- 기반 배치: 비용 최적화를 위한 환경 설정
- Amazon EC2 스팟 인스턴스 개요

지속 가능성

지속 가능성 원칙은 AWS 워크로드의 지속 가능성 목표를 달성하는 데 도움이 되는 설계 원칙, 운영 지침, 모범 사례 및 개선 계획을 제공합니다.

구현에 대한 구현 지침은 지속 가능성 원칙 - AWS Well-Architected Framework 백서에서 확인할 수 있습니다.

중점 영역

- 설계 원칙
- 리전 선택
- 수요에 맞춘 조정
- 소프트웨어 및 아키텍처
- 데이터 관리
- 하드웨어 및 서비스
- 리소스

설계 원칙

게임 산업의 지속 가능성은 전 세계 여러 리전에서 다양한 속도로 진화하고 있습니다. 북미의 대규모 지역에서 지속 가능한 전력과 EU 및 영국의 지속 가능성 요구 사항을 통해 아키텍트는 가까운 시일 내에 더 그린 워크로드를 달성하기 위한 여러 접근 방식을 취합니다. Well-Architected 지속 가능성 원칙을 사용하여 다양한 워크로드에 대해 이러한 조치를 달성할 수 있습니다.

렌즈의이 섹션에서는 게임 워크로드에 사용할 수 있는 몇 가지 모범 사례를 설명합니다.

- 게임 사용자 데이터에 적합한 스토리지 유형을 선택합니다.
- 데이터 중복을 제거하고 워크로드에서 불필요한 데이터를 제거하는 데이터 수명 주기 정책에 유의하세요.
- 컴퓨팅 리소스 배포의 크기를 적절하게 조정할 수 있습니다.
- 간략 및 트랜잭션 프로세스에 서버리스를 사용합니다.

리전 선택

Games Lens와 관련된 리전 선택 모범 사례는 없습니다. 자세한 내용은 지속 가능성 원칙 - AWS Well-Architected Framework를 참조하세요.

수요에 맞춘 조정

Games Lens와 관련된 수요 모범 사례에는 부합되지 않습니다. 자세한 내용은 지속 가능성 원칙 - AWS Well-Architected Framework를 참조하세요.

소프트웨어 및 아키텍처

Games Lens와 관련된 소프트웨어 및 아키텍처 모범 사례는 없습니다. 자세한 내용은 지속 가능성 원칙 - AWS Well-Architected Framework를 참조하세요.

데이터 관리

GAMESUS01: 게임 시스템에서 사용자 및 게임 데이터를 어떻게 관리하나요?

중요한 기록 데이터만 유지하여 데이터 스토리지와 관련성을 최적화하는 데이터 수명 주기 전략을 개발합니다.

모범 사례

- GAMESUS01-BP01 사용자 콘텐츠, 구독자 정보 및 게임 내 구매에 맞게 조정된 패턴에 맞는 스토리지 기술 사용
- GAMESUS01-BP02 수명 주기 정책 또는 TTL 만료를 사용하여 불필요한 게임 사용자 데이터, 로그 파일 또는 더 이상 사용되지 않는 자산 삭제

GAMESUS01-BP01 사용자 콘텐츠, 구독자 정보 및 게임 내 구매에 맞게 조정된 패턴에 맞는 스토리지 기술 사용

유형, 보존 요구 사항 및 액세스 빈도별로 데이터를 분류해야 합니다. 이를 통해 게임 또는 백엔드 서비스에서 생성하는 다양한 데이터 유형에 가장 최적화된 스토리지 솔루션을 선택할 수 있습니다. 빠르게 변경되는 데이터는 키 값 또는 인 메모리 데이터베이스 서비스에 저장해야 합니다. 트랜잭션 데이터는

관계형 데이터베이스 서비스에 저장해야 합니다. 대용량 파일, 게임 자산 또는 사용자 생성 콘텐츠는 객체 스토리지 서비스에 저장해야 합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

게임은 액세스 빈도, 지연 시간 및 비용에 최적화된 스토리지 솔루션이 필요한 다양한 데이터 유형을 생성하고 사용합니다. 저장된 데이터는 태그를 사용하여 분류하여 제거할 수 있거나 장기간 저장해야 하는 데이터를 구분해야 합니다.

다음 서비스는 다양한 게임 사용 사례에 적합합니다.

Amazon Aurora(MySQL 및 PostgreSQL과 호환)는 고가용성, 저지연 및 자동 조정을 제공하므로 플레이어 계정 관리 및 인증, 게임 내 경제, 리더보드 및 플레이어 순위, 게임 상태 지속성, 이벤트 및 캠페인 관리, 다중 리전 및 고가용성 배포와 같은 대량의 트랜잭션 데이터를 처리하는 데 매우 적합합니다.

Amazon DynamoDB는 지연 시간이 짧고 처리량이 높으며 확장성이 원활한 것으로 알려진 완전관리형 NoSQL 데이터베이스로, 실시간 플레이어 데이터, 세션 관리, 인벤토리, 게임 내 경제, 실시간 멀티플레이어 게임 상태, 매치메이킹, 이벤트 로깅 및 글로벌 시청자를 위한 규모 조정을 처리하는 데 이상적입니다.

Amazon DocumentDB(MongoDB와 호환)는 확장 가능하고 지연 시간이 짧은 문서 지향 데이터베이스 서비스를 제공하며, 인벤토리 시스템, 플레이어 프로필 및 사용자 지정, 게임 월드 및 절차적으로 생성된 콘텐츠, 소셜 및 플레이어 상호 작용, 분석 및 동작 추적, 게임 내 메타데이터 및 구성과 같은 유연한 반정형 데이터를 저장하는 데 적합합니다.

Amazon ElastiCache는 Redis 또는 Memcached를 사용한 인 메모리 캐싱을 지원하여 빠른 데이터 액세스와 응답 시간 단축을 제공합니다. 이는 원활한 사용자 경험을 위해 속도와 성능이 필수적인 실시간 멀티플레이어 게임에 매우 중요합니다. ElastiCache는 실시간 리더보드, 세션 관리, 게임 메타데이터 캐싱, 게임 내 채팅 및 메시징, 매치메이킹, 실시간 분석 및 원격 측정, 트래픽이 많은 이벤트 규모 조정에 게임에 사용됩니다.

Amazon Simple Storage Service(S3)를 사용하여 게임 자산, 비디오, 사진, 텍스트 로그 파일 등과 같은 객체를 저장할 수 있습니다. S3는 업계 최고의 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다.

빈번하고 빈도가 낮은 데이터 액세스를 지원하는 여러 스토리지 클래스와 비용 효율적인 아카이브 스토리지를 제공하는 경우. 개발 전반에 걸쳐 자주 액세스하는 데이터의 경우 스튜디오는 짧은 지연 시간과 높은 처리량 성능을 위해 S3 Standard에 객체를 저장해야 합니다. 자주 핫에서 콜드로 또는 그 반대로 이동하는 데이터의 경우 스튜디오는 S3 Intelligent-Tiering을 조사해야 합니다. Intelligent-Tiering은

데이터의 액세스 패턴을 모니터링하고 데이터를 가장 비용 효율적인 액세스 계층으로 자동으로 이동합니다.

높은 처리량과 짧은 지연 시간이 필요하고 단일 가용 영역에 상주해도 괜찮은 스튜디오의 경우 S3 Express One Zone을 사용합니다. 이렇게 하면 데이터를 단일 AZ에 복제하고 S3 표준에 비해 데이터 액세스 속도를 개선할 수 있습니다. 기록 데이터의 심층 아카이브 요구 사항을 위해 Amazon은 Amazon Glacier도 제공합니다. Amazon Glacier 스토리지 클래스는 데이터 아카이빙을 위해 특별히 구축되어 클라우드에서 고성능, 검색 유연성 및 저비용 아카이브 스토리지를 제공합니다.

Amazon Elastic Block Store를 사용하여 게임 서버 또는 자산 리포지토리가 작동하는 데 필요한 게임 서버의 바이너리, 실행 파일 및 구성을 저장할 수 있습니다. EC2 인스턴스에 연결되지 않은 미사용 볼륨을 스냅샷 처리하고 삭제해야 합니다. 이렇게 하면 불필요한 서비스 및 하드웨어의 사용량을 줄이면서 발생하는 스토리지 요금이 줄어듭니다.

구현 단계

- 게임 데이터를 유형, 보존 요구 사항 및 액세스 빈도별로 분류하고 데이터에 태그를 지정하여 단기 스토리지 요구 사항과 장기 스토리지 요구 사항을 구분합니다.
- 트랜잭션 데이터에 Amazon Aurora를 사용하고, 실시간 플레이어 데이터에 DynamoDB를 사용하고, 반정형 데이터에 DocumentDB를 사용하고, 시간에 중요한 게임 정보의 지연 시간이 짧은 캐싱에 ElastiCache를 사용합니다.
- Amazon S3에 게임 자산, 로그 및 사용자 생성 콘텐츠를 저장하고, 액세스 패턴 및 아카이브 요구 사항에 따라 적절한 스토리지 클래스(예: Intelligent-Tiering, One Zone 및 Glacier)를 선택하고, 정기적인 스냅샷 관리를 통해 게임 서버 바이너리 및 구성에 EBS를 사용합니다.

GAMESUS01-BP02 수명 주기 정책 또는 TTL 만료를 사용하여 불필요한 게임 사용자 데이터, 로그 파일 또는 더 이상 사용되지 않는 자산 삭제

태그와 데이터 유형을 사용하여 수명 주기 정책 또는 TTL을 생성하여 데이터를 아카이브 스토리지로 이동하거나 서비스에서 완전히 제거할 수 있습니다. 여기에는 임시 구성, 만료된 아카이브된 콘텐츠 및 더 이상 필요하지 않은 기록 로그가 포함될 수 있습니다. 대부분의 서비스는 태그 지정을 지원합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

S3에 저장된 데이터의 경우 수명 주기 정책을 사용하여 데이터를 자주 액세스하지 않는 스토리지 계층 및 아카이브 계층으로 이동할 수 있습니다. S3 수명 주기 구성에서, 객체를 한 스토리지 클래스에

서 다른 스토리지 클래스로 전환하여 스토리지 비용을 절약하도록 규칙을 정의할 수 있습니다. 객체의 액세스 패턴을 모르거나 시간이 지남에 따라 액세스 패턴이 변하면 자동 비용 절감을 위해 객체를 S3 Intelligent-Tiering 스토리지 클래스로 전환할 수 있습니다.

Amazon S3은 다음 다이어그램과 같이 스토리지 클래스 간 전환을 위한 폭포형(Waterfall) 모델을 지원합니다.

S3 수명 주기 구성에 이전 작업을 추가하여 Amazon S3가 수명 주기 종료 시 객체를 삭제하도록 지시할 수 있습니다. 수명 주기 구성에 따라 객체의 수명이 종료되면 Amazon S3는 버킷이 있는 S3 버전 관리 상태에 따라 만료 작업을 수행합니다.

- 버전이 지정되지 않은 버킷: Amazon S3는 제거를 위해 객체를 대기열에 넣고 비동기적으로 제거하여 객체를 영구적으로 제거합니다.
- 버전 관리가 활성화된 버킷: 현재 객체 버전이 삭제 마커가 아닌 경우 Amazon S3는 고유한 버전 ID로 삭제 마커를 추가합니다. 이를 통해 최신 버전이 최신이 아닌 버전이 되고 삭제 마커가 최신 버전이 됩니다.
- 버전 관리가 일시 중지된 버킷: Amazon S3는 null을 버전 ID로 사용하여 삭제 마커를 생성합니다. 이 삭제 마커는 객체를 효과적으로 삭제하는 버전 계층 구조에서 객체 버전을 null 버전 ID로 대체합니다.
- 버킷에 수명 주기 구성을 추가할 경우 구성 규칙이 기존 객체는 물론 나중에 추가하는 객체에도 적용됩니다. 예를 들어 특정 접두사가 있는 객체가 생성 후 30일 후에 만료되도록 하는 만료 작업과 함께 수명 주기 구성 규칙을 오늘 추가하는 경우 Amazon S3는 30일 이상 경과하고 지정된 접두사가 있는 기존 객체를 제거하기 위해 대기열에 넣습니다.

DynamoDB용 Time To Live(TTL)는 더 이상 관련이 없는 항목을 삭제하기 위한 비용 효율적인 방법입니다. TTL을 사용하면 항목이 더 이상 필요하지 않은 시점을 나타내는 항목별 만료 타임스탬프를 정의할 수 있습니다. DynamoDB는 쓰기 처리량을 소비하지 않고 만료 후 며칠 내에 만료된 항목을 자동으로 삭제합니다.

- TTL을 사용하려면 먼저 테이블에서 TTL을 활성화한 다음 TTL 만료 타임스탬프를 저장할 특정 속성을 정의해야 합니다. 타임스탬프는 초 단위로 Unix epoch 시간 형식으로 저장해야 합니다. 항목이 생성되거나 업데이트될 때마다 만료 시간을 계산하여 TTL 속성에 저장할 수 있습니다.
- 유효하고 만료된 TTL 속성이 있는 항목은 일반적으로 만료 후 며칠 이내에 시스템에서 삭제할 수 있습니다. 삭제 보류 중인 만료된 항목은 여전히 TTL 속성 변경 또는 제거 등의 작업으로 업데이트할 수 있습니다. 만료된 항목을 업데이트할 때는 조건식을 사용하여 해당 항목이 이후에 삭제되지 않도록 하는 것이 좋습니다. 필터 표현식을 사용하여 스캔 및 쿼리 결과에서 만료된 항목을 제거합니다.

- 삭제된 항목은 일반적인 삭제 작업을 통해 삭제된 항목과 비슷하게 작동합니다. 삭제되면 항목은 사용자 삭제 대신 서비스 삭제로 DynamoDB Streams로 이동하고 다른 삭제 작업과 마찬가지로 로컬 보조 인덱스 및 글로벌 보조 인덱스에서 제거됩니다.

ElastiCache for Redis를 사용하면 TTLs 또는 캐시된 키의 만료를 사용하여 캐시된 데이터의 최신성을 제어할 수 있습니다. 설정된 시간이 지나면 키가 캐시에서 삭제되고 업데이트된 데이터에 도달하는 동시에 오리진 데이터 스토어에 대한 액세스가 필요합니다.

- 두 가지 원칙에 따라 적용할 적절한 TTLs과 구현할 캐싱 패턴 유형이 결정됩니다. 먼저 기본 데이터의 변경 속도를 이해하는 것이 중요합니다. 둘째, 업데이트된 데이터 대신 오래된 데이터가 애플리케이션에 반환될 위험을 평가하는 것이 중요합니다.
- 동적 데이터가 자주 변경되면 기본 데이터베이스의 변경 속도와 일치하는 변경 속도로 데이터를 만료시키는 더 낮은 TTLs을 적용할 수 있습니다. 이렇게 하면 데이터베이스 요청을 오프로드하기 위한 버퍼를 제공하면서 오래된 데이터를 반환할 위험이 줄어듭니다.
- 또한 몇 분 또는 몇 초 동안만 데이터를 캐싱하고 지속 시간이 길더라도 캐시된 키에 TTLs을 적절하게 적용하면 성능이 향상되고 게임에서 전반적으로 더 나은 플레이어 경험을 얻을 수 있다는 점을 인식해야 합니다.

구현 단계

- Amazon S3 수명 주기 정책을 사용하여 객체를 빈번하지 않은 액세스 또는 아카이브 계층으로 전환하고 수명 주기 규칙에 따라 불필요한 객체를 삭제하도록 만료 작업을 구성합니다.
- DynamoDB 테이블에서 TTL(Time to Live)을 활성화하여 쓰기 처리량을 소비하지 않고 만료된 항목을 자동으로 삭제하여 Unix epoch 시간으로 만료 타임스탬프를 정의합니다.
- 데이터 변경 속도 및 오래된 데이터에 대한 위험 허용치를 기반으로 ElastiCache 키에 적합한 TTLs을 설정하여 캐시된 데이터 신선도를 높이고 플레이어 경험을 개선합니다.

하드웨어 및 서비스

GAMESUS02: 게임 백엔드에서 컴퓨팅 리소스 사용을 어떻게 관리하나요?

Studio는 다양한 컴퓨팅 유형, 관리형 서비스 및 절감형 플랜을 혼합하여 사용량을 최적화하는 컴퓨팅 전략을 개발해야 합니다. 또한 게임 서버와 백엔드 서비스가 컴퓨팅 인스턴스에 패키징되는 방식을 최적화하여 불필요한 리소스 수를 줄여야 합니다.

모범 사례

- GAMESUS02-BP01 적절한 컴퓨팅 워크로드에 적합한 관리형 서비스 선택
- GAMESUS02-BP02 필요한 경우에만 컴퓨팅 크기 조정 및 GPU 성능 배포

GAMESUS02-BP01 적절한 컴퓨팅 워크로드에 적합한 관리형 서비스 선택

이벤트 기반 또는 매우 가변적인 트래픽 워크로드에 관리형 서비스를 사용하도록 게임 백엔드 서비스를 설계합니다. 관리형 서비스는 인프라 관리를 로 전환 AWS 하고 다중 테넌트 컨트롤 플레인으로 인해 여러 사용자에게 환경에 미치는 영향을 분산합니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

AWS AWS Lambda AWS Fargate (컨테이너) 및 Amazon Gamelift(게임 서버 오케스트레이션)와 같은 서비스는 기본 인프라를 관리할 필요 없이 코드, 컨테이너를 실행하거나 게임 서버를 오케스트레이션할 수 있습니다. 이러한 서비스는 플레이어 수요에 따라 자동으로 조정되며 사용하는 리소스에 대해서만 요금이 부과됩니다. 기본 인프라는 사용자를 대신하여 관리되므로 게임 및 백엔드 서비스의 요구 사항에만 집중할 수 있습니다.

AWS Lambda를 사용하면 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다. Lambda는 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버 및 운영 체제 유지 관리, 용량 프로비저닝 및 자동 조정, 로깅을 포함한 컴퓨팅 리소스 관리를 수행합니다. Lambda를 사용하면 Lambda가 지원하는 언어 런타임 중 하나로 코드를 제공해야 합니다. Lambda는 게임 이벤트, 플레이어 인증, 게임 내 구매 처리 및 매치메이킹 요청을 처리하는 데 유용합니다. Lambda는 이벤트 수에 따라 자동으로 규모를 조정하며 예상치 못한 트래픽 급증을 처리할 수 있습니다.

AWS Fargate는 Amazon Elastic Container Service(ECS) 및 Amazon Elastic Kubernetes Service(EKS)와 함께 작동하는 컨테이너를 위한 서버리스 컴퓨팅 엔진입니다. AWS Fargate 는 서버를 프로비저닝하고 관리할 필요성을 완화하여 애플리케이션 구축에 쉽게 집중할 수 있도록 하고, 애플리케이션당 리소스를 지정하고 비용을 지불할 수 있도록 하며, 애플리케이션 격리를 통해 설계를 통해 보안을 개선합니다. Fargate는 플레이어 프로필, 상태 관리 및 매치메이킹을 처리하는 백엔드 서비스에 적합합니다.

Amazon GameLift는 세션 기반 멀티플레이어 게임을 위한 전용 게임 서버를 배포, 운영 및 확장하기 위한 관리형 서비스입니다. 단 몇 분 만에 클라우드에 첫 번째 게임 서버를 배포하여 선결제 소프트웨어 개발에서 최대 수천 시간의 엔지니어링 시간을 절약하고 개발자가 설계에서 멀티플레이어 기능을 잘라낼 수 있는 기술적 위험을 줄일 수 있습니다.

구현 단계

- 자동 조정 및 서버리스 관리를 활용하여 게임 이벤트, 플레이어 인증, 게임 내 구매, 매치메이킹 요청 처리와 같은 이벤트 기반 워크로드 AWS Lambda 에를 사용합니다.
- 플레이어 프로필, 상태 관리 및 매치메이킹과 같은 백엔드 서비스를 위해 ECS 또는 EKS와 AWS Fargate 함께 배포하여 서버 관리를 제거하고 애플리케이션 격리를 개선합니다.
- Amazon GameLift를 사용하여 세션 기반 멀티플레이어 게임을 위한 전용 게임 서버를 배포하고 확장하여 개발 시간과 운영 복잡성을 줄일 수 있습니다.

GAMESUS02-BP02 필요한 경우에만 컴퓨팅 크기 조정 및 GPU 성능 배포

컴퓨팅 리소스를 효율적으로 활용하도록 게임 서버와 백엔드를 설계합니다. 컴퓨팅을 과도하게 프로비저닝하면 불필요한 비용이 발생할 수 있으며 유휴 리소스 또는 사용률이 낮은 리소스의 양을 최소화할 수 있습니다. GPU 인스턴스는 Unreal에서 HLOD 재구축과 같은 특정 개발 작업을 지원하는 데 사용하거나 게임 서버에 설계상 필요한 경우 사용해야 합니다. 이렇게 하면 워크로드의 환경 영향과 비용이 크게 줄어듭니다.

이 모범 사례가 확립되지 않을 경우 노출되는 위험 수준: 높음

구현 지침

여러 EC2 인스턴스 유형과 필요한 최소 수의 인스턴스를 사용하도록 게임 서버와 백엔드 서비스를 최적화해야 합니다. 이렇게 하면 개발 중 또는 게임 시작 시 요구 사항을 충족하는 데 사용할 수 있는 인스턴스 수가 늘어납니다. 또한 인스턴스 유형을 배포 중인 특정 워크로드와 일치해야 합니다. 컴퓨팅 최적화 인스턴스는 게임 서버 및 매치메이킹과 같은 백엔드 서비스를 비롯한 다양한 사용 사례를 지원합니다. 메모리 최적화 인스턴스는 메모리의 대용량 데이터 세트를 처리하는 워크로드에 빠른 성능을 제공하도록 설계되었습니다. 고성능 요구 사항에 필요한 GPU 인스턴스를 사용하지만 일반 컴퓨팅 작업에는 사용하지 않습니다. 가능하면 AWS Graviton 인스턴스를 사용하여 ARM에서 실행할 서비스 또는 게임 서버를 설계합니다. Graviton은에서 사용할 수 있는 에너지 효율적인 인스턴스 유형 중 가장 성능이 좋습니다 AWS. 또한 x86 인스턴스 유형에 비해 향상된 성능과 비용을 제공합니다.

AWS Compute Optimizer를 사용하여 Amazon Elastic Compute Cloud(EC2) 인스턴스 유형, Amazon Elastic Block Store(EBS) 볼륨 구성,의 Amazon Elastic Container Service(ECS) 서비스의 작업 크

기 AWS Fargate, 상용 소프트웨어 라이선스, AWS Lambda 함수 메모리 크기, Amazon Relational Database Service(RDS) DB 인스턴스 클래스와 같은 최적의 AWS 리소스 구성을 식별하고 기계 학습을 사용하여 과거 사용률 지표를 분석합니다. Compute Optimizer는 워크로드에 최적의 AWS 리소스를 권장하여 비용을 절감하고 AWS 워크로드 성능을 높이는 일련의 APIs와 콘솔 환경을 제공합니다.

구현 단계

- 게임 서버용 컴퓨팅 최적화 인스턴스, 대규모 데이터 세트용 메모리 최적화 인스턴스, HLOD 재구축 또는 GPU 종속 게임 서버와 같은 작업에만 GPU 인스턴스를 사용하여 컴퓨팅 리소스를 특정 워크로드와 일치시킵니다.
- x86 인스턴스에 비해 에너지 효율성, 성능 향상 및 비용 절감을 위해 가능한 경우 AWS Graviton 인스턴스를 배포하여 컴퓨팅 사용률을 최적화합니다.
- AWS Compute Optimizer 를 사용하여 과거 사용률을 분석하고 EC2, AWS ECS AWS Lambda및 Amazon RDS 워크로드에 가장 효율적인 구성을 권장하여 비용을 절감하고 성능을 개선합니다.

리소스

지속 가능성과 관련된 모범 사례에 대해 자세히 알아보려면 다음 리소스를 참조하세요.

- UN: 게임 산업, 지구에 대한 위협 강조
- 중간: 게임 개발의 환경 지속 가능성: 보다 환경친화적인 미래를 위해 책임감 있게 재생

주요 AWS 서비스

- Amazon Aurora
- Amazon DynamoDB
- Amazon DocumentDB(MongoDB 호환)
- Amazon ElastiCache
- Amazon S3
- Amazon S3 스토리지 클래스
- Amazon S3 Intelligent-Tiering 스토리지 클래스
- Amazon S3 Express One Zone 스토리지 클래스
- Amazon Elastic Block Store
- AWS Lambda

- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service:
- Amazon GameLift
- AWS Compute Optimizer

결론

게임은 플레이어의 전 세계 시청자에게 엔터테인먼트 경험을 제공하고 일반적으로 예측할 수 없고 가변적인 사용 특성을 갖도록 설계되었습니다. Games Industry Lens는 일반적으로 게임 아키텍처를 구성하는 일반적인 유형의 시나리오를 설명하고 클라우드에서 게임을 구축하고 운영할 때 고려해야 할 일련의 질문과 모범 사례를 제공합니다. 이 프레임워크를 게임 아키텍처에 적용하면 클라우드에서 안정적이고 안전하며 효율적이고 비용 효율적인 게임을 구축할 수 있습니다.

기여자

다음 개인이 이 문서에 기여했습니다.

- Adam Hatfield, Amazon Web Services의 Senior Solutions Architect
- Brady Webb, Amazon Web Services의 기술 계정 관리자
- Bruce Ross - Amazon Web Services의 Well-Architected Lens Leader, Senior Solutions Architect
- Caleb Cecil, Amazon Web Services의 Associate Solutions Architect
- Carlos Perez, 클라우드 최적화 성공 SA, Amazon Web Services
- Chase Herrington, Amazon Web Services의 ESL 기술 계정 관리자.
- Chris Blackwell, Amazon Web Services의 선임 솔루션 아키텍트
- Corey Ouderkirk, Amazon Web Services의 선임 기술 계정 관리자
- Derek Villavicencio, Prototyping SA, Amazon Web Services
- Erik Ynigo Becerril, Amazon Web Services의 Senior Solutions Architect
- Grzegorz Ochmanski, Amazon Web Services의 Sr. Solutions Architect
- Hadrian Baron, Amazon Web Services의 Sr. Technical Account Manager
- Ian Armbruster, Amazon Web Services의 Sr. Customer Solutions Manager
- Jed Obray, Amazon Web Services의 선임 기술 계정 관리자
- Khurram Khokhar, Amazon Web Services의 선임 기술 계정 관리자
- Kyle Somers, Amazon Web Services 솔루션 아키텍처의 Sr. Manager
- Madhuri Srinivasan, Amazon Web Services Well-Architected의 Senior Technical Writer
- Matthew Wygant, Amazon Web Services Well-Architected의 Sr. TPM 지침
- Nataliya Godunok, Amazon Web Services의 Cloud Optimization Success SA
- Nirav Doshi, Amazon Web Services의 Principal Solutions Architect
- Olivia Liddell, Amazon Web Services의 솔루션 아키텍트
- Randy James, Amazon Web Services의 Principal Technical Account Manager
- Reou Ando, Amazon Web Services의 게임 솔루션 아키텍트
- Richard Raseley, Amazon Web Services의 선임 기술 계정 관리자
- Sam Patzer, Amazon Web Services의 Senior Solutions Architect
- Scott Selinger, Amazon Web Services의 Sr. Solutions Architect
- Sean Allen, Amazon Web Services의 Sr. Solutions Architect

- Serge Poueme, Amazon Web Services의 Senior Solutions Architect
- Stewart Matzek, Amazon Web Services Well-Architected의 Senior Technical Writer
- Trenton Potgieter, Amazon Web Services의 선임 솔루션 아키텍트 AI/ML/Analytics

문서 수정

이 백서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
새 렌즈 버전	전체 렌즈가 새로운 모범 사례 지침으로 업데이트되었습니다.	2025년 12월 9일
최초 게시	백서가 처음 게시되었습니다.	2021년 11월 19일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.