AWS Whitepaper

# Lensa Industri Game

# Lensa Industri Game: AWS Whitepaper

# Table of Contents

# Lensa Industri Game — Kerangka AWS Well-Architected

Tanggal publikasi: 9 Desember 2025 () Revisi dokumen

AWS Well-Architected Framework membantu arsitek cloud membangun infrastruktur yang aman, berkinerja tinggi, tangguh, dan efisien untuk aplikasi dan beban kerja mereka. Berdasarkan enam pilar — keunggulan operasional, keamanan, keandalan, efisiensi kinerja, optimalisasi biaya, dan keberlanjutan — Well-Architected memberikan pendekatan yang konsisten bagi pelanggan AWS dan Mitra untuk mengevaluasi arsitektur, memulihkan risiko, dan menerapkan desain yang memberikan nilai bisnis.

Dalam lensa ini, kami fokus pada cara merancang, menyebarkan, dan merancang beban kerja game Anda di. AWS Cloud Kami mendefinisikan komponen, mengeksplorasi skenario beban kerja umum, dan menguraikan prinsip-prinsip desain yang membantu Anda menerapkan Kerangka Kerja Well-Architected. Kami menyarankan Anda mulai merancang arsitektur Anda dengan mempertimbangkan praktik dan pertanyaan terbaik dari whitepaper AWS Well-Architected Framework. Dokumen ini memberikan praktik terbaik tambahan untuk pelanggan industri game.

Lensa ini menentukan praktik terbaik yang dimaksudkan untuk mengatasi karakteristik unik membangun dan mengoperasikan game di cloud berdasarkan pengalaman kami bekerja dengan pengembang dan penerbit industri game di seluruh dunia. Ini memberikan panduan tentang cara merancang dan mengoperasikan lingkungan Anda sehingga biaya dioptimalkan dan dapat diskalakan untuk fluktuasi permintaan pemain global. Lensa ini juga memberikan panduan untuk mengamankan infrastruktur game dan kinerja tuning Anda untuk memberikan pengalaman pemain yang positif.

Dokumen ini ditujukan bagi mereka yang berperan dalam teknologi, seperti chief technology officer (CTOs), direktur teknis studio game, arsitek, pengembang, dan anggota tim operasi. Setelah membaca dokumen ini, Anda akan memahami praktik dan strategi AWS terbaik untuk digunakan saat merancang arsitektur untuk game.

## Ketersediaan lensa

Lensa khusus memperluas panduan praktik terbaik yang disediakan oleh AWS Well-Architected Tool. AWS WA Tool memungkinkan Anda untuk membuat lensa kustom Anda sendiri, atau menggunakan lensa yang dibuat oleh orang lain yang telah dibagikan dengan Anda.

Untuk mulai meninjau beban kerja game Anda, unduh dan impor Lensa Industri Game AWS Well-Architected Tool dari repositori lensa kustom AWS Well-Architected publik. GitHub

# Prinsip desain

AWS Well-Architected Framework mengidentifikasi prinsip-prinsip desain umum berikut untuk memfasilitasi desain yang baik di cloud untuk beban kerja game:

- Memahami perilaku pemain dan pola penggunaan untuk mengembangkan permainan dan membantu melindungi pemain: Untuk mendorong peningkatan terus-menerus ke permainan Anda dan mengelola pengalaman pemain secara efektif, penting untuk memiliki visibilitas tentang bagaimana pemain berinteraksi dengan game itu sendiri dan dengan pemain lainnya. Ini membantu Anda memahami cara meningkatkan permainan, mengelola biaya, dan memantau serta bereaksi terhadap aktivitas penggunaan yang tidak sah yang menimbulkan risiko bagi pengalaman pemain.
- Gunakan teknologi yang menyederhanakan operasi game dan meningkatkan kecepatan pengembangan: Prioritaskan adopsi teknologi yang dapat meningkatkan kecepatan dan mengurangi biaya operasional untuk menghadirkan fitur dan peningkatan baru kepada pemain. Permainan digerakkan oleh hit dan ada banyak pilihan bagi pemain untuk dipertimbangkan, jadi bergerak cepat dan beradaptasi dengan perubahan sangat penting untuk keberhasilan permainan. Pertimbangkan apakah Anda merasa nyaman mengoperasikan perangkat lunak Anda sendiri atau jika Anda lebih suka mengadopsi layanan terkelola yang sebanding dari AWS, AWS Mitra, atau keduanya.
- Optimalkan arsitektur Anda untuk meningkatkan metrik yang mencerminkan pengalaman pemain yang sebenarnya: Saat Anda beradaptasi dan mengembangkan arsitektur Anda dari waktu ke waktu, pikirkan bagaimana peningkatan dan perubahan ini akan memengaruhi pengalaman pemain. Beban kerja game harus mampu menahan dan meminimalkan dampak kegagalan untuk memblokir gangguan gameplay yang meluas. Fitur dan sistem permainan yang tidak terlalu bergantung satu sama lain harus di-de-coupled untuk mengurangi dampak kegagalan dan mengisolasi masalah yang berdampak pada pemain.
- Merancang infrastruktur untuk memenuhi konkurensi pemain puncak dan skala dinamis sesuai kebutuhan: Infrastruktur harus dirancang untuk skala untuk memenuhi permintaan pemain. Metrik, seperti konkurensi sesi pemain dan jumlah login, dapat digunakan untuk skala terlebih dahulu sebelum sistem menjadi kelebihan beban. Metrik pemanfaatan sistem reaktif, seperti CPU dan konsumsi memori, dapat digunakan untuk skala setelah sistem menjadi kelebihan beban. Dengan menskalakan infrastruktur Anda secara dinamis, Anda dapat mengurangi biaya pengoperasian game Anda.

- Menerapkan runbook untuk meningkatkan operasi game: Runbook operasional harus digunakan untuk mengelola tugas operasi game berulang secara konsisten. Runbook harus ada untuk alur kerja operasi game umum seperti menyelidiki dan menanggapi laporan pemain, mengelola aktivitas pra-penskalaan infrastruktur untuk mempersiapkan acara skala besar yang diantisipasi seperti peluncuran musim baru dan rilis konten game, dan untuk mengatasi aktivitas pemeliharaan game yang khas.

# Skenario

Di bagian ini, kami membahas beberapa skenario yang umum dalam arsitektur game. Setiap skenario mencakup karakteristik umum yang mendorong desain dan contoh diagram arsitektur referensi.

Skenario

- Hosting game untuk gameplay sinkron waktu nyata
- Backend permainan
- Arsitektur backend game berbasis tanpa server
- Pengembangan game cloud (CGD)
- Pipa analitik game

## Hosting game untuk gameplay sinkron waktu nyata

Gameplay sinkron real-time memungkinkan dua atau lebih pemain untuk berpartisipasi dan berinteraksi dalam permainan secara bersamaan di mana keadaan gameplay dibagi antara pemain yang terhubung untuk menciptakan sedekat mungkin dengan pengalaman real-time. Contoh game sinkron termasuk first-person shooter, game online multiplayer massively (MMOG), game olahraga dan aksi, atau game online di mana dua atau lebih pemain harus terhubung untuk berbagi pengalaman bermain dalam waktu dekat.

Karakteristik arsitektur gameplay sinkron real-time meliputi:

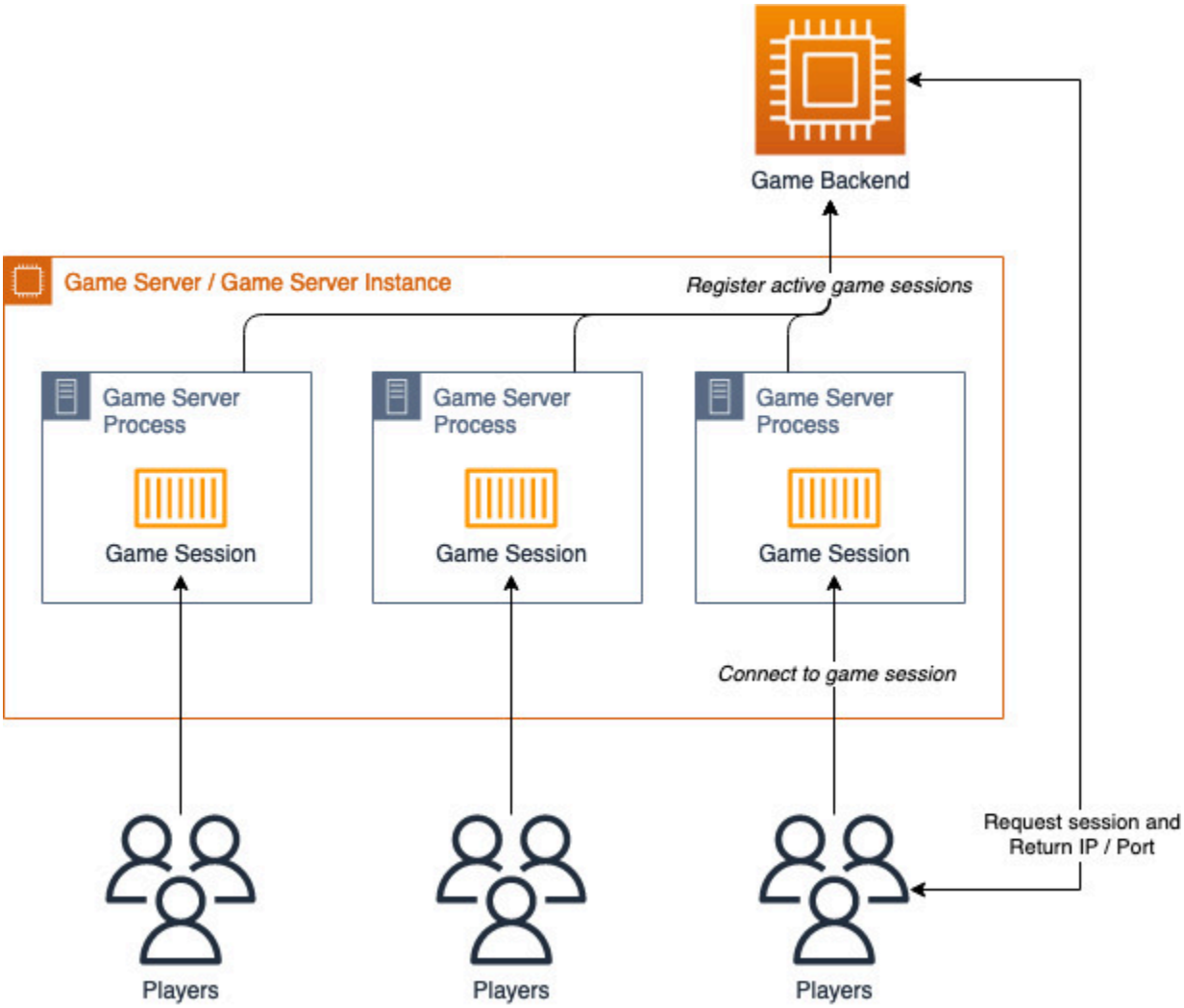
- Beberapa game dapat di-host sebagai sesi game melalui proses server game yang berjalan di server khusus. Beberapa game mungkin menggunakan arsitektur seperti P2P yang menggunakan Session Traversal Utilities yang lebih ringan untuk NAT (STUN) atau Traversal Using Relay di sekitar server NAT (TURN). Terlepas dari jenis server yang terlibat, server game di-host di beberapa pusat data dan Wilayah AWS secara global.
- Klien game dapat bergabung dengan sesi permainan baik dengan meminta kecocokan dari layanan perjodohan terpusat yang dihosting di sistem backend game atau dengan memilih kecocokan dari daftar server game yang tersedia yang telah ditentukan sebelumnya. Klien game dilengkapi dengan alamat IP dan port untuk terhubung.
- Banyak game sinkron sensitif terhadap latensi, seperti penembak orang pertama dan game online multipemain masif. Mereka kemungkinan akan menyertakan algoritme seperti mundur dan

pelebaran waktu untuk meminimalkan efek lag tetapi mungkin juga memiliki toleransi latensi yang telah ditentukan sebelumnya yang diukur dan dioptimalkan dengan cermat untuk mengurangi pengalaman lag yang terkadang dapat terjadi pada pemain dalam situasi latensi tinggi. Informasi latensi ini ditentukan dengan menginstrumentasi klien game untuk melakukan ping ke server game yang tersedia Wilayah AWS untuk menangkap metrik seperti latensi, jitter jaringan, dan metrik penting lainnya untuk pengalaman bermain game. Metrik ini dikirim ke layanan pengumpulan metrik pusat di sistem backend game sehingga streaming operasi langsung dapat memantau kesehatan game. Selama proses perjodohan, klien game memberikan data latensi mereka saat ini sebagai salah satu parameter permintaan saat meminta kecocokan, dan layanan perjodohan dapat menggunakan data latensi tersebut sebagai salah satu variabel saat memilih server game untuk meng-host pemain.

- Biasanya, gameplay dilakukan melalui campuran protokol (seperti server game yang menggunakan pesan berbasis UDP yang lebih cepat dengan perjodohan, otentikasi, dan lalu lintas client-server lainnya menggunakan HTTPS).
- Server game menggunakan algoritme dan desain untuk meminimalkan lalu lintas client-server seperti streaming transformasi, delta, dan kompresi data.
- Server game sering menjadi target aktivitas jahat dan harus dilindungi dengan solusi perlindungan DDo S seperti AWS Shield Advanced.

## Proses server game

Diagram berikut menggambarkan arsitektur server game yang khas. Ini menggambarkan hubungan logis antara instance server game dan proses server game yang menjadi tuan rumah sesi game.



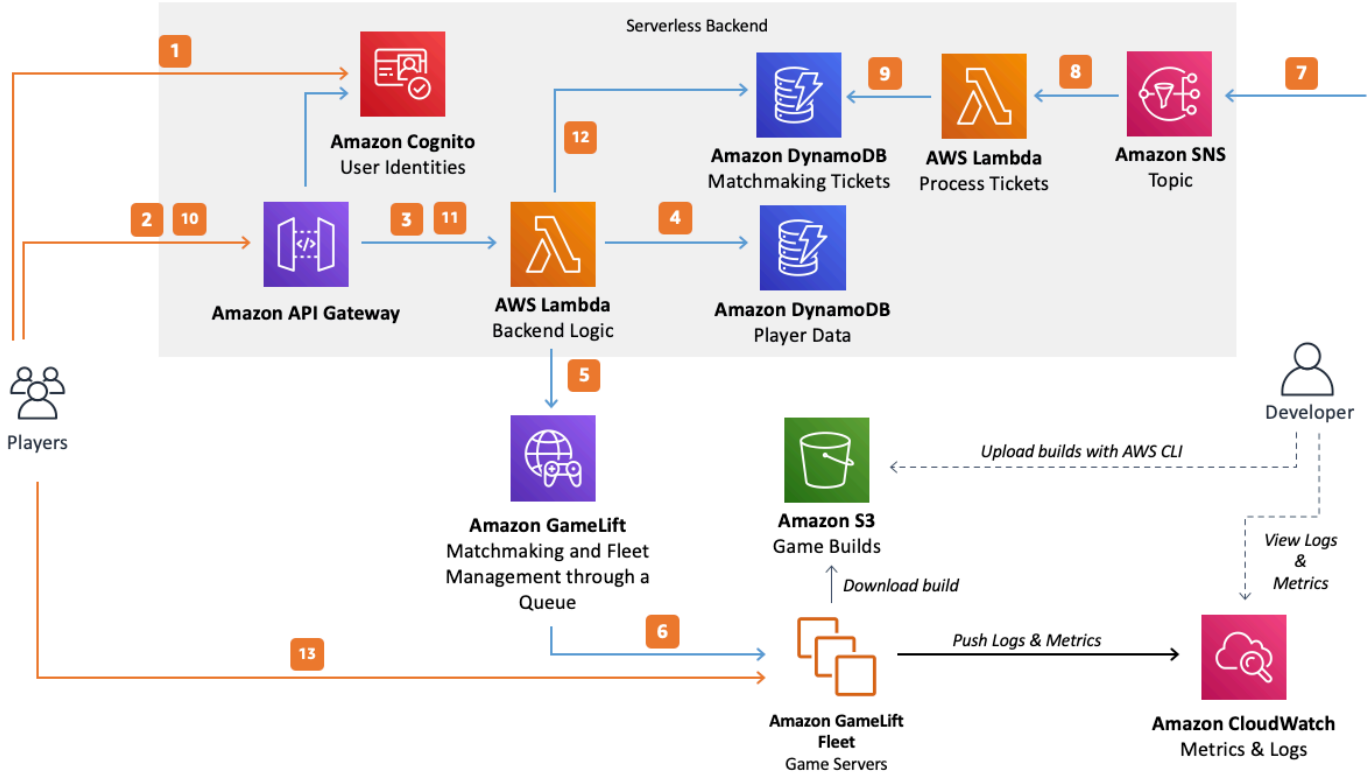

Arsitektur server game logis

- EC2 Instans Amazon digunakan sebagai server game, juga dikenal sebagai instance server game. Server game menghosting satu atau lebih proses server game, dan masing-masing menjalankan salinan build server game Anda. Biasanya, beberapa proses server game berjalan pada instance server game untuk memanfaatkan sumber daya komputasi secara efisien dan mengurangi biaya. Ketika sesi permainan aktif dan siap untuk menjadi tuan rumah sesi pemain, statusnya diperbarui dengan backend game (biasanya layanan perjodohan) sehingga dapat mulai digunakan untuk menjadi tuan rumah pemain.
- Backend game dapat memberi pemain alamat IP dan port server yang menjadi tuan rumah sesi permainan sehingga mereka dapat terhubung untuk bermain.

# Hosting server game berbasis sesi dengan backend tanpa server

Saat mengembangkan arsitektur untuk game Anda, pertimbangkan fitur dan kemampuan yang Anda butuhkan dan tingkat overhead manajemen operasional yang siap Anda miliki. Untuk memberikan keseimbangan terbaik antara kemudahan pengoperasian dan fleksibilitas, Anda dapat membangun game menggunakan layanan terkelola dari penyedia cloud. Layanan terkelola memberi Anda kontrol untuk mengembangkan dan menyesuaikan fitur game kustom Anda sendiri, sekaligus mengurangi beban Anda untuk menerapkan dan mengelola infrastruktur.

Hosting game multipemain berbasis sesi membutuhkan infrastruktur server untuk meng-host proses server game serta backend yang dapat diskalakan untuk perjodohan dan manajemen sesi. Arsitektur referensi berikut menunjukkan bagaimana Amazon GameLift mengelola hosting dan backend tanpa server dapat digunakan untuk mengelola game berbasis sesi Anda.




Amazon GameLift mengelola hosting untuk game berbasis sesi

Diagram menjelaskan proses memasukkan pemain ke dalam game yang berjalan di hosting game yang GameLift dikelola. Ini mencakup langkah-langkah berikut:

1. Klien game meminta identitas Amazon Cognito dari kumpulan identitas Amazon Cognito. Ini secara opsional dapat dihubungkan ke penyedia identitas eksternal.

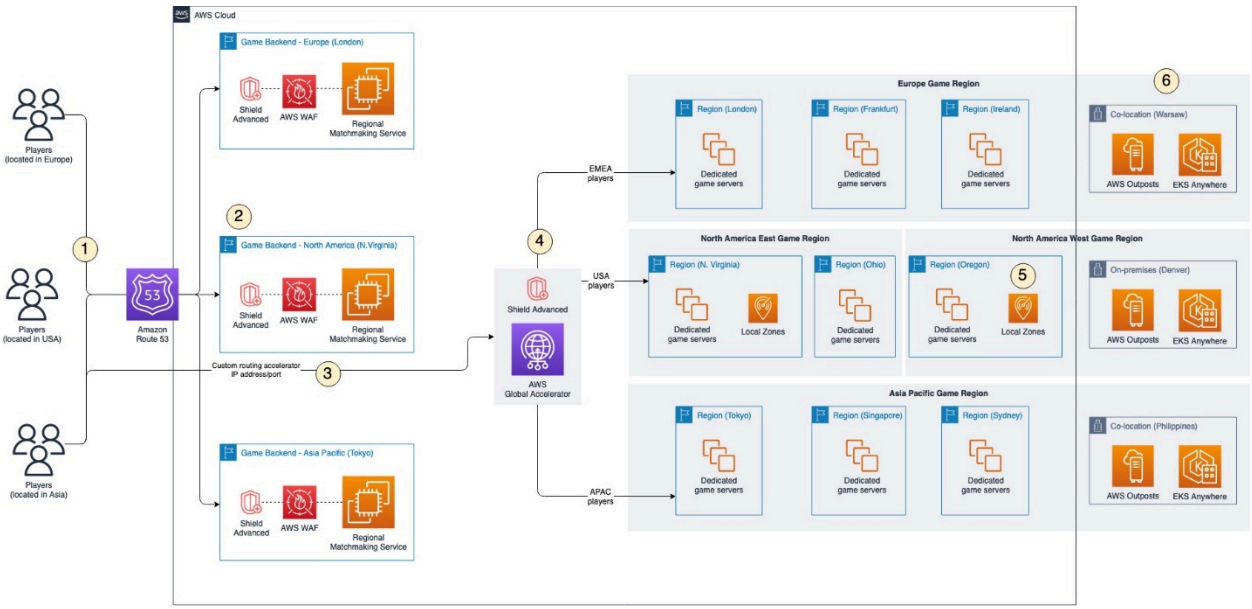
2. Klien game menerima kredensil akses sementara dan meminta sesi game melalui Amazon API Gateway dengan menandatangani permintaan dengan kredenal Amazon Cognito.
3. API Gateway memanggil AWS Lambda fungsi.
4. Fungsi Lambda meminta data pemutar dari tabel Amazon DynamoDB. Identitas Amazon Cognito digunakan untuk meminta data pemain yang benar dengan aman karena identitas yang diautentikasi disediakan dalam data konteks permintaan.
5. Menggunakan data pemain yang benar untuk informasi tambahan (seperti tingkat keterampilan pemain), fungsi Lambda meminta pertandingan melalui GameLift FlexMatch perjodohan. Anda dapat menentukan konfigurasi FlexMatch perjodohan dengan dokumen konfigurasi berbasis JSON. Klien game dapat menghasilkan metrik latensi dengan melakukan ping titik akhir server di berbagai Wilayah, dan data latensi dapat digunakan untuk mendukung perjodohan berbasis latensi.
6. Setelah FlexMatch mencocokkan kelompok pemain yang cocok dengan latensi yang sesuai ke Wilayah, ia meminta penempatan sesi permainan melalui GameLift antrian. Antrian berisi armada dengan satu atau lebih lokasi Wilayah terdaftar.
7. Ketika sesi ditempatkan di salah satu lokasi armada, notifikasi kejadian dikirim ke topik Amazon SNS.
8. Sebuah fungsi Lambda akan menerima kejadian Amazon SNS dan memprosesnya.
9. Jika pesan Amazon SNS adalah MatchmakingSucceeded peristiwa, fungsi Lambda menulis hasilnya ke DynamoDB dengan port server dan alamat IP. Nilai time-to-live (TTL) digunakan untuk memastikan bahwa tiket perjodohan dihapus dari DynamoDB ketika mereka tidak lagi diperlukan.
10. Client game membuat permintaan yang telah ditandatangani ke API Gateway untuk memeriksa status tiket matchmaking pada interval tertentu.
11. API Gateway memanggil fungsi Lambda yang memeriksa status tiket matchmaking.
12. Fungsi Lambda memeriksa DynamoDB untuk menentukan apakah tiket telah berhasil. Jika telah berhasil, fungsi Lambda mengirimkan alamat IP, port, dan ID sesi pemain kembali ke client. Jika tiket gagal, fungsi Lambda mengirimkan respons yang menyatakan bahwa kecocokan belum siap.
13. Client game menghubungkan ke server game menggunakan TCP atau UDP dengan menggunakan port dan alamat IP yang disediakan oleh backend. Ini mengirimkan ID sesi pemain ke server game, dan server game memvalidasinya menggunakan Amazon GameLift Server SDK.

Atau, Anda dapat memodifikasi arsitektur sebelumnya untuk menggunakan API Gateway dengan WebSockets Amazon. GameLift Dalam pendekatan ini, komunikasi antara klien game dan layanan backend game Anda terjadi menggunakan implementasi WebSocketberbasis. Implementasi ini

dapat digunakan sehingga fungsi Lambda backend game memulai pesan sisi server ke klien game daripada menerapkan model polling. WebSocket

## Multi-Region dan arsitektur hybrid untuk game latensi rendah

Bagian ini menjelaskan arsitektur Multi-region dan hybrid untuk game latensi rendah.




Mengurangi latensi dengan akselerasi jaringan dan server game yang digunakan secara global

1. Pemain dalam game yang tersedia secara global dapat berasal dari mana saja. Ketika seorang pemain meminta sesi permainan atau pertandingan, klien game mereka mengirimkan permintaan ke layanan backend game yang terdaftar di Amazon Route 53. Routing berbasis latensi Route 53 dapat digunakan untuk mengarahkan pemain ke backend game terdekat yang tersedia.
2. Backend game digunakan di beberapa populasi yang Wilayah AWS paling dekat dengan pemain. Setiap backend game mencakup layanan perjodohan Regional yang menemukan sesi permainan dari seluruh Wilayah game. Meskipun permintaan perjodohan pemain diproses oleh layanan perjodohan Regional di dekat mereka, layanan perjodohan mampu mengarahkan pemain ke sesi permainan di Wilayah permainan lain, jika perlu. Tindakan ini meningkatkan ketahanan dan kinerja. Selain itu, setiap layanan backend game menggunakan AWS WAF dan AWS Shield Advanced menyediakan penyaringan web layer-7 dan kontrol bot, dan perlindungan terhadap serangan penolakan distribusi layanan (S). DDo Anda memiliki banyak opsi untuk bagaimana Anda membangun layanan backend game Anda, seperti tanpa server, kontainer, EC2 instance, atau hosting layanan backend game di pusat data Anda sendiri.

3. Untuk meningkatkan pengalaman bagi pemain dengan mengurangi latensi jaringan dan jitter, akselerator perutean khusus digunakan menggunakan AWS Global Accelerator, yang secara otomatis mengoptimalkan perutean lalu lintas dari klien game ke server game menggunakan jaringan global. AWS Anda mengonfigurasi akselerator perutean khusus untuk memetakan port pendengar Akselerator Global ke port instans server game Anda. EC2 Klien game terhubung ke IP dan port Global Accelerator yang berfungsi sebagai proxy dan secara deterministik mengarahkan pemain ke IP server game dan port yang benar yang menjadi tuan rumah sesi permainan.
4. Game Anda mencakup Wilayah permainan logis ramah pemain yang mewakili kumpulan lokasi hosting server game yang secara geografis dekat satu sama lain, seperti Amerika Utara atau Asia Pasifik. Untuk mengurangi latensi dan meningkatkan cakupan geografis, kombinasi berbagai solusi hosting server game dapat digunakan untuk meningkatkan pengalaman pemain. Prioritaskan penggunaan Wilayah AWS jika memungkinkan, karena lokasi ini berfitur lengkap dan berisi jejak kapasitas terbesar.
5. Gunakan AWS Local Zones untuk meng-host server game di lokasi geografis pemain yang kurang terlayani di mana Anda tidak memiliki fasilitas hosting yang ada, atau tidak Wilayah AWS tersedia.
6. Terapkan AWS Outposts ke pusat data lokal yang ada dan penyedia lokasi bersama untuk menciptakan bidang kontrol dan pengalaman manajemen yang mulus di setiap lokasi penerapan menggunakan rak yang dikelola sepenuhnya dan server yang dipasang di rak. AWS Outposts juga bermanfaat jika Anda tidak memiliki kapasitas server yang ada di lingkungan lokal Anda. Namun, jika Anda menginginkan implementasi hybrid dengan perangkat lunak yang berjalan pada infrastruktur server Anda sendiri yang ada, Anda harus menggunakan Amazon EKS Anywhere, yang memungkinkan Anda membuat dan menjalankan cluster Kubernetes di infrastruktur Anda sendiri dengan konektivitas kembali ke Amazon EKS. Ini memberikan tampilan konsol yang konsisten dari klaster Kubernetes Anda di lokal.

# Backend permainan

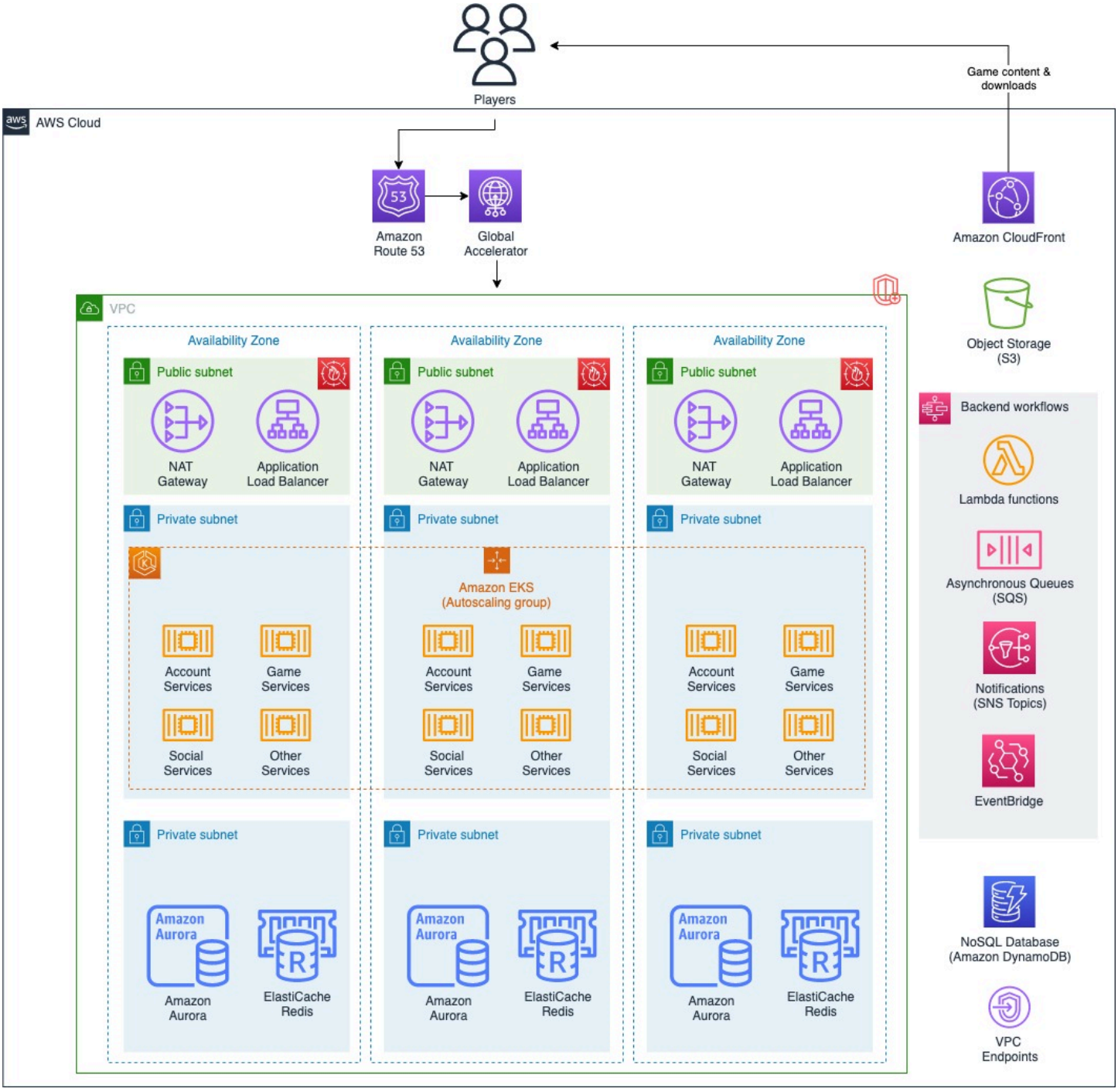
Backend game digunakan untuk mengelola permainan dan status pemain, serta mengintegrasikan fitur sosial dan tingkat sistem ke dalam game yang mendukung pengalaman bermain game. Manajemen profil pemain, penyimpanan item dan inventaris, serta statistik dan papan peringkat adalah contoh layanan yang dihosting di backend game.

Backend game biasanya dibangun sebagai REST APIs yang diakses oleh klien menggunakan HTTPS. Namun, pendekatan lain juga umum, seperti WebSockets yang menyediakan saluran dua arah untuk kasus penggunaan seperti pemberitahuan klien untuk obrolan dan kehadiran dalam

game. Backend game dapat digunakan menggunakan berbagai arsitektur penerapan yang berbeda, termasuk menggunakan instance, kontainer, atau arsitektur tanpa server.

## Arsitektur backend game berbasis kontainer

Bagian ini menguraikan arsitektur backend game berbasis container.



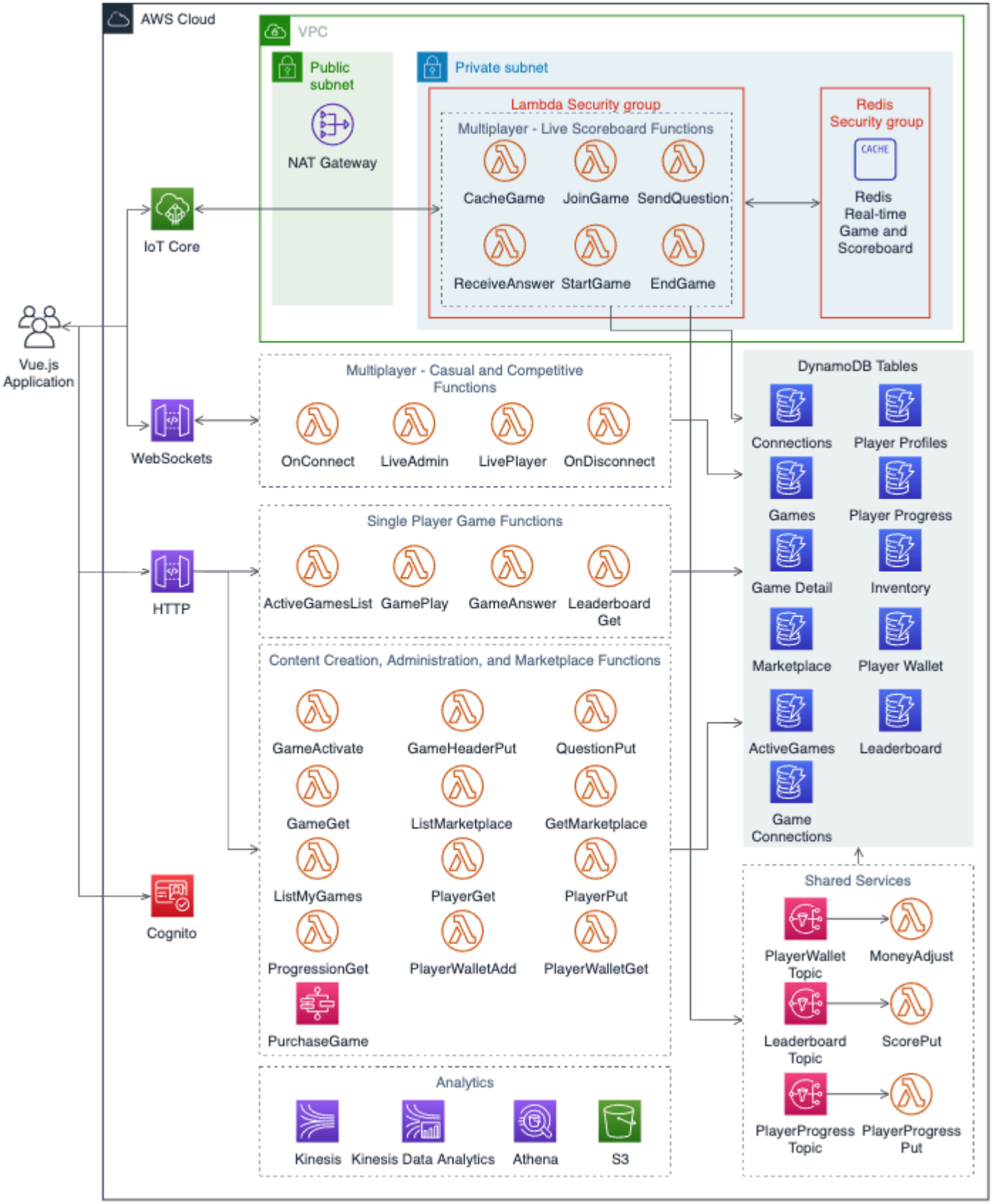

Hosting backend game menggunakan wadah

- Pemain mengakses game menggunakan perangkat lunak klien game, yang dapat didistribusikan kepada mereka melalui sistem game, etalase digital, atau unduhan langsung dari jaringan pengiriman konten (CDN), seperti Amazon. CloudFront CDNsmenyediakan caching di lokasi tepi untuk mempercepat kinerja bagi pengguna yang mengunduh konten. Misalnya, CloudFront dapat digunakan untuk mendistribusikan perangkat lunak klien game ke pemain Anda serta aset game dan konten lainnya.
- AWS Global Accelerator menyediakan akselerasi lalu lintas dan kontrol yang dapat disesuaikan untuk merutekan lalu lintas dari klien game pemain ke penyeimbang beban Anda serta merutekan lalu lintas lintas Wilayah untuk tujuan Multi-wilayah dan failover. Nama domain khusus untuk REST backend game Anda APIs dikonfigurasi di Amazon Route 53 untuk merutekan lalu lintas ke titik akhir Global Accelerator. Tidak ditampilkan dalam diagram,AWS Shield Advanced dapat memberikan mitigasi DDo S tambahan untuk akselerator dan backend game Anda.
- NAT Gateway dan Application Load Balancer dikerahkan ke subnet publik di setiap Availability Zone yang digunakan oleh backend game untuk menyediakan ketersediaan tinggi dengan Wilayah.AWS WAF digunakan pada Application Load Balancer untuk menyediakan penyaringan lalu lintas web layer-7.
- Backend game di-host sebagai kumpulan layanan mikro berbasis kontainer individu yang digunakan ke cluster Amazon EKS di subnet pribadi yang tersebar di Availability Zone untuk ketahanan. Penskalaan otomatis secara dinamis menyesuaikan kapasitas layanan dan node cluster berdasarkan pemanfaatan sumber daya, yang biasanya berkorelasi dengan permintaan pemain. Sedangkan Cluster Autoscaler secara otomatis menyesuaikan jumlah node di cluster, Horizontal Pod Autoscaler secara otomatis menskalakan pod yang di-deploy ke dalam cluster.
- Data game dan pemain disimpan dalam database backend dan cache yang digunakan ke subnet pribadi di seluruh Availability Zones dengan replikasi antara node primer dan replika. Amazon Aurora adalah pilihan populer untuk kasus penggunaan seperti profil pemain, hak, dan pembelian dalam game, yang dapat memiliki persyaratan kueri yang lebih kompleks dan dapat memanfaatkan kemampuan pemodelan data relasional MySQL dan PostgreSQL. Amazon ElastiCache berguna untuk membangun papan peringkat berkinerja tinggi, pub/sub perpesanan, dan untuk menyimpan data yang sering diakses untuk mengurangi latensi dan beban pada database. Amazon DynamoDB adalah penyimpanan data NoSQL yang dikelola sepenuhnya yang ideal untuk pola akses yang tidak dapat diprediksi dan kemampuan untuk menskalakan ke throughput yang hampir tidak terbatas untuk kasus penggunaan seperti data status pemain dan game, data sesi, inventaris dan penyimpanan item, atau kasus penggunaan di mana Anda menginginkan database global dengan overhead minimal.

- Alur kerja pemrosesan asinkron harus digunakan untuk melakukan pekerjaan yang dapat dilakukan di latar belakang, seperti memperbarui papan peringkat atau mengirim permintaan pertemanan. Konfigurasikan backend game Anda untuk mendorong jenis pekerjaan ini ke dalam antrian Amazon SQS agar sesuai skala saat game Anda tumbuh atau pertimbangkan untuk menggunakan topik Amazon SNS untuk mendistribusikan pekerjaan di antara banyak antrian aplikasi konsumen untuk pemrosesan paralel. Gunakan fungsi AWS Lambda untuk melakukan pemrosesan dengan cara yang didorong oleh peristiwa untuk mengurangi biaya infrastruktur komputasi dan overhead manajemen Anda. Untuk alur kerja yang berumur panjang atau memerlukan koordinasi tugas dengan beberapa langkah, pertimbangkan untuk mengatur seluruh alur kerja menggunakan.AWS Step Functions Amazon EventBridge dapat digunakan untuk memulai fungsi untuk menanggapi AWS layanan dan peristiwa aplikasi kustom.

# Arsitektur backend game berbasis tanpa server

Banyak pengembang game tidak ingin mengelola infrastruktur, dan sebaliknya lebih suka membangun game mereka menggunakan teknologi yang memungkinkan mereka untuk fokus pada perangkat lunak. Arsitektur tanpa server direkomendasikan dalam skenario ini karena memungkinkan Anda untuk membangun dan merilis fitur lebih cepat, dan dengan overhead operasional yang lebih sedikit. Arsitektur tanpa server dirancang menggunakan layanan cloud yang dapat menskalakan secara dinamis berdasarkan permintaan tanpa perlu mengatur, mengelola, dan menskalakan server. Arsitektur referensi berikut menggambarkan cara membangun game menggunakan arsitektur tanpa server.


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

Arsitektur referensi backend game berbasis tanpa server

Arsitektur referensi ini menggambarkan permainan trivia berbasis web yang menyediakan fitur pemain tunggal dan multipemain.

- Otentikasi pemain: Pemain mengautentikasi menggunakan Amazon Cognito, yang menyediakan otentikasi aman dengan direktori pengguna untuk manajemen identitas pemain.
- Logika permainan sebagai fungsi tanpa server: Fitur game dan logika bisnis backend dijalankan sebagai AWS Lambda fungsi yang dimulai sebagai respons terhadap peristiwa, yang menekan biaya karena Anda hanya membayar saat fungsi berjalan. Lambda memberi Anda fleksibilitas untuk menulis setiap fitur game sebagai layanan mikro terpisah menggunakan bahasa pemrograman pilihan Anda. Misalnya, Anda dapat memilih untuk mengembangkan fungsi.NET Lambda jika Anda memiliki pengalaman menggunakan C# untuk membangun game Unity, atau Anda dapat memilih untuk mengembangkan fungsi Lambda Node.js jika Anda ingin memprogram frontend dan backend untuk game berbasis web keduanya di. JavaScript
- NoSQL Data Store untuk data game dan pemain: Gunakan DynamoDB untuk menyimpan data pemain dan game Anda, karena ini dibuat khusus untuk menyimpan sejumlah besar data dari layanan mikro. Seperti yang diilustrasikan dalam arsitektur ini, ini adalah praktik terbaik untuk menggunakan penyimpanan data terpisah untuk setiap kebutuhan penyimpanan data fitur game, yang membuatnya mudah bagi Anda untuk memantau dan mengelola fitur secara mandiri. Ini juga menciptakan batasan pemisahan jika kepemilikan fitur atau layanan berubah dalam tim Anda. Dalam arsitektur referensi ini, tabel DynamoDB digunakan untuk menyimpan data seperti status koneksi, detail permainan, kemajuan pemain, dan informasi papan peringkat.
- Gameplay pemain tunggal: Fitur pemain tunggal memungkinkan pemain untuk melakukan tindakan seperti memilih dan memainkan game dan melihat papan peringkat. Fitur-fitur ini diimplementasikan sebagai layanan RESTful backend yang dihosting dengan API HTTP Amazon API Gateway yang memanggil fungsi Lambda yang sesuai untuk mendapatkan dan mengatur data dalam tabel DynamoDB. Saat gameplay selesai, backend juga mengirimkan notifikasi ke topik Amazon SNS yang memulai fungsi Lambda secara asinkron untuk menyimpan kemajuan dan statistik pemain.
- Gameplay multipemain: Fitur game multipemain mengharuskan pemain untuk dapat berinteraksi dengan game untuk point-to-point komunikasi, serta menyiarkan dan menerima pembaruan dari pemain lain yang terhubung. WebSockets Implementasi cocok untuk point-to-point komunikasi dalam game ringan, seperti trivia. Pemain dapat membuat WebSockets koneksi ke Amazon API Gateway WebSockets, yang mengelola koneksi dan hanya memanggil fungsi Lambda ketika ada pesan untuk dikirim atau diterima untuk pemain. Untuk kasus penggunaan di mana one-to-

many komunikasi diperlukan antara pemain, AWS IoT Core berikan dukungan untuk pengiriman pesan menggunakan WebSockets melalui MQTT, yang memungkinkan klien untuk berlangganan topik dan bertindak berdasarkan pesan yang diterimanya. Dalam arsitektur ini, WebSockets over MQTT digunakan untuk mendukung kasus penggunaan seperti menyiarkan pembaruan langsung dalam game dan mengajukan pertanyaan kepada pemain yang terhubung. Sebagai alternatif AWS IoT, Anda dapat memilih Redis Pub/Sub untuk pengiriman pesan atau Redis Streams jika Anda memerlukan penyimpanan pesan.

- Gunakan fungsi Lambda berkemampuan VPC untuk mengakses sumber daya di subnet pribadi Anda: Konfigurasikan fungsi Lambda berkemampuan VPC untuk mengakses sumber daya di subnet pribadi VPC Anda, seperti Amazon, yang digunakan untuk mengurangi waktu kueri untuk kumpulan data latensi ElastiCache rendah seperti papan peringkat langsung.

Untuk informasi selengkapnya, lihat Panduan untuk Hosting Backend Game Kustom di. AWS

## Pengembangan game cloud (CGD)

Cloud game development (CGD) mengacu pada infrastruktur dan alat yang diperlukan untuk siklus hidup pengembangan game untuk membangun, menguji, dan mengembangkan game. Pengembangan game bersifat kolaboratif antara pengguna dan persyaratan infrastruktur sering berubah selama tahap pengembangan.

Banyak pengembang game merangkul tim pengembangan yang didistribusikan secara global dan jarak jauh, yang membutuhkan teknologi yang mendukung jenis pengembangan ini. Pengembang game dapat meng-host semua atau sebagian dari lingkungan ini AWS dan menggunakan ketersediaan global Wilayah AWS untuk menempatkan sumber daya lebih dekat dengan pengguna dan mengelola lingkungan pengembangan mereka dengan lebih hemat biaya dengan menskalakan komputasi dan penyimpanan sesuai kebutuhan.

Lingkungan dapat bervariasi tergantung pada kebutuhan pengembang game, tetapi mereka biasanya mencakup workstation pengembang untuk seniman, desainer, insinyur, penguji QA, kontraktor, dan personel lain untuk melakukan pekerjaan mereka. Lingkungan ini juga biasanya mencakup build farm yang terdiri dari repositori kode sumber bagi pengguna untuk memeriksa perubahan mereka dan CI/CD infrastruktur untuk membangun, mengemas, dan menguji artefak yang dikembangkan.

Arsitektur produksi game ini memiliki karakteristik sebagai berikut:

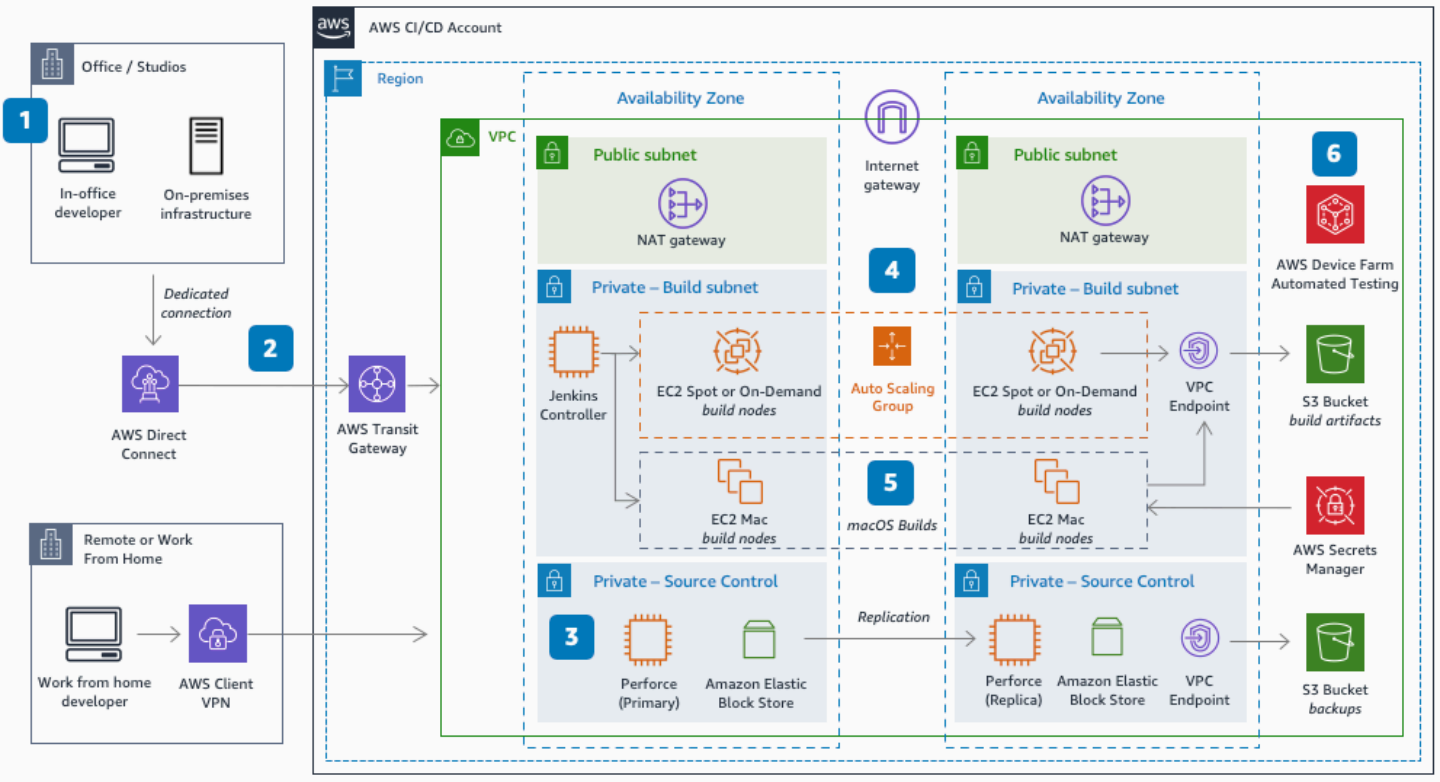
- Pengguna harus dapat mengakses workstation virtual melalui browser web atau klien desktop lokal, seperti Amazon DCV, yang memberi mereka sesi streaming latensi rendah untuk mengakses

perangkat lunak dan alat yang sama yang akan mereka akses jika mereka bekerja pada mesin di kantor atau studio pengembangan. Workstation virtual ini, biasanya server berbasis cloud, harus memungkinkan pengguna untuk berkolaborasi dan mengerjakan proyek mereka sepenuhnya di lingkungan cloud melalui LAN atau WAN. Ketika pengguna tidak aktif menggunakan mesin, pekerjaan mereka harus dicadangkan ke penyimpanan cloud yang tahan lama, misalnya repositori kontrol sumber atau sistem file seperti Amazon Elastic File System (EFS) dan Amazon FSx, dan mesin mereka harus dimatikan untuk mengurangi biaya.

- Repositori kontrol sumber, seperti Perforce, harus dirancang dengan ketersediaan tinggi menggunakan replikasi antara Availability Zone atau antara lingkungan lokal dengan backup yang disimpan di penyimpanan cloud seperti Amazon S3. Misalnya, server Perforce berbasis cloud harus menyertakan server komit utama yang dihosting di satu Availability Zone dengan replikasi ke server siaga yang dihosting di Availability Zone lain di Region yang sama.
- Sumber daya pengembangan game build farm harus dirancang dengan penskalaan otomatis sehingga sumber daya komputasi disediakan sesuai kebutuhan, dan Instans EC2 Spot harus digunakan untuk mengurangi biaya yang dikeluarkan saat mengukur jumlah server yang diperlukan untuk build.

## Pengembangan game cloud: CI/CD

CI/CD Infrastruktur penting saat mengembangkan game terlepas dari ukuran tim untuk meningkatkan waktu iterasi, membangun keandalan, penerapan yang efisien, dan kontrol yang lebih baik atas proses pengembangan dan rilis untuk memberikan pengalaman game berkualitas tinggi kepada pemain. CI/CD Pipa pengembangan game biasanya terdiri dari server dan penyimpanan kontrol sumber yang sangat tersedia, sumber daya komputasi untuk menjalankan build Anda, dan perangkat lunak untuk melakukan pengujian otomatis, bersama dengan konektivitas jaringan yang tepat dari mesin pengembangan Anda. Arsitektur referensi berikut menunjukkan cara membongkar build game dari lingkungan pengembangan game jarak jauh atau lokal ke AWS Cloud untuk membantu pengembang dalam memigrasi atau membangun build farm baru.




Game offload dibangun ke cloud

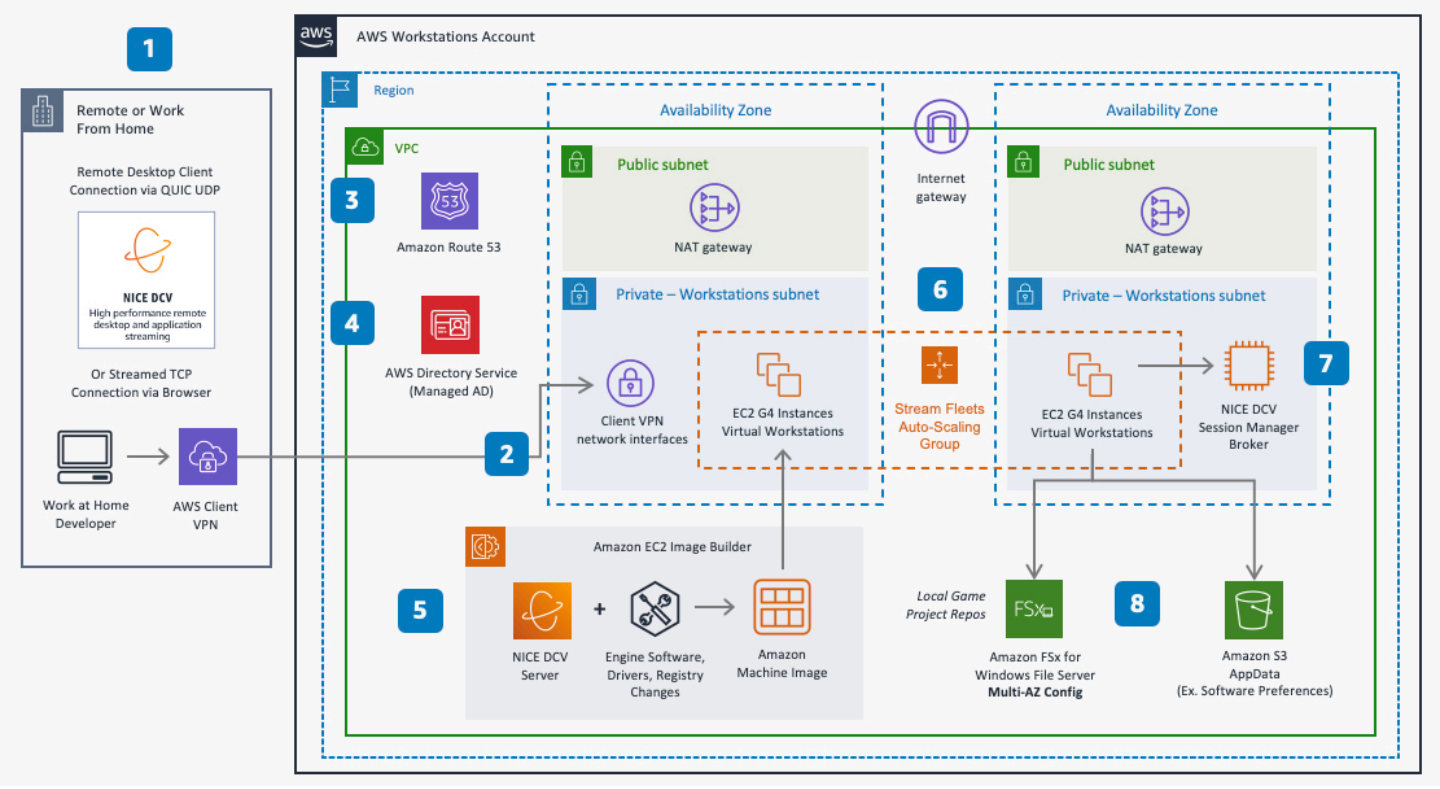
1. AWS Direct Connect menyediakan koneksi latensi rendah, pribadi, khusus AWS untuk pengembang di kantor. Pengembang jarak jauh menggunakan teknologi tanpa kepercayaan seperti Akses Terverifikasi AWS, atau jaringan pribadi virtual (VPN) seperti Client VPN AWS .
2. AWS Transit Gateway menyederhanakan manajemen jaringan untuk konektivitas antara VPCs dan dari jaringan lokal.
3. Perforce mengelola kontrol sumber dan versi (CI) yang didukung oleh penyimpanan Amazon EBS untuk data persisten yang diakses dengan cepat. Perforce Helix Core tersedia di.AWS Marketplace
4. Komit memulai build (CD) di Jenkins ketika pengembang mendorong perubahan ke Perforce yang diikat ke cabang. Perforce memulai POST muatan JSON ke Jenkins. Pengontrol Jenkins memanggil perintah CLI tanpa kepala engine untuk menjalankan dan memparalelkan proses pembuatan di seluruh node Docker sementara (seperti Instans Spot Amazon atau Instans On-Demand Amazon). EC2 EC2 Pengembang dapat meningkatkan ketersediaan dengan menggunakan dua pengendali Jenkins, satu di setiap Availability Zone, di belakang penyeimbang beban. Untuk beberapa mesin game, pengembang mungkin memerlukan infrastruktur lisensi tambahan yang dikonfigurasi dalam subnet tambahan untuk menjual lisensi untuk konteks build setiap kali build bersamaan dijalankan.

5. Bagian Xcode dari build iOS diturunkan ke instans Amazon EC2 Mac untuk menandatangani, membangun, dan mengekspor file.IPA, memisahkan proses dan mengurangi waktu pembuatan.AWS Secrets Manager memegang profil penyediaan, kunci pribadi, dan sertifikat.
6. Artefak build dikirim ke Amazon S3, yang mengirimkan pemberitahuan keberhasilan atau kegagalan. AWS Device Farm memungkinkan pengujian otomatis untuk perangkat seluler.

## Pengembangan game cloud: Workstation

Pengembangan game sering melibatkan tim terdistribusi yang bekerja dari jarak jauh dari berbagai lokasi, membutuhkan akses ke infrastruktur bersama dan kemampuan untuk mendukung pengembangan kolaboratif yang tersebar secara geografis. Tren menuju proses pengembangan game yang lebih terdistribusi ini, termasuk penggunaan work-for-hire studio dan pekerjaan jarak jauh, mengharuskan penerapan teknologi dan alur kerja yang kuat untuk memfasilitasi produktivitas, keahlian bersama, dan praktik pengembangan yang gesit.

Arsitektur referensi berikut menunjukkan cara menggunakan AWS untuk hosting workstation pengembangan game jarak jauh menggunakan protokol Amazon DCV.




Streaming pengembangan game dari mana saja dengan Amazon DCV

1. Amazon DCV adalah protokol streaming yang mendukung streaming 4K, 60-FPS. Pengembang menggunakan browser terhubung melalui koneksi TCP, sedangkan klien desktop dapat menggunakan QUIC UDP melalui port 8443 untuk meningkatkan kinerja.
2. Pengembang menggunakan AWS Client VPN untuk koneksi aman ke antarmuka jaringan di subnet workstation dengan terjemahan alamat jaringan sumber (SNAT).
3. Amazon Route 53 menyediakan DNS pribadi untuk sumber daya di VPC serta penerusan DNS masuk dan keluar.
4. Directory Service menyediakan Microsoft Active Directory terkelola untuk mengaktifkan penyimpanan proyek game lokal yang dipetakan ke pengguna individu.
5. Workstation dibuat menggunakan Amazon Machine Image (AMI) yang dibangun dengan Image Builder. Gambar termasuk Amazon DCV Server, perangkat lunak pengembang, perubahan registri, dan driver seperti driver game NVIDIA atau driver periferal. AWS Marketplace termasuk umum AMIs digunakan untuk workstation.
6. Armada workstation menggunakan tipe EC2 instans Amazon grafis yang menyediakan GPUs, dan diskalakan menggunakan grup Auto Scaling. EC2
7. Broker Manajer Sesi Amazon DCV memungkinkan pengelolaan sesi Amazon DCV.
8. Penyimpanan file lokal proyek di-host di Amazon FSx untuk Windows File Server. Pengembang berkomitmen pada pipa CI/CD terpisah dengan mendorong dari penyimpanan lokal ke kontrol sumber.

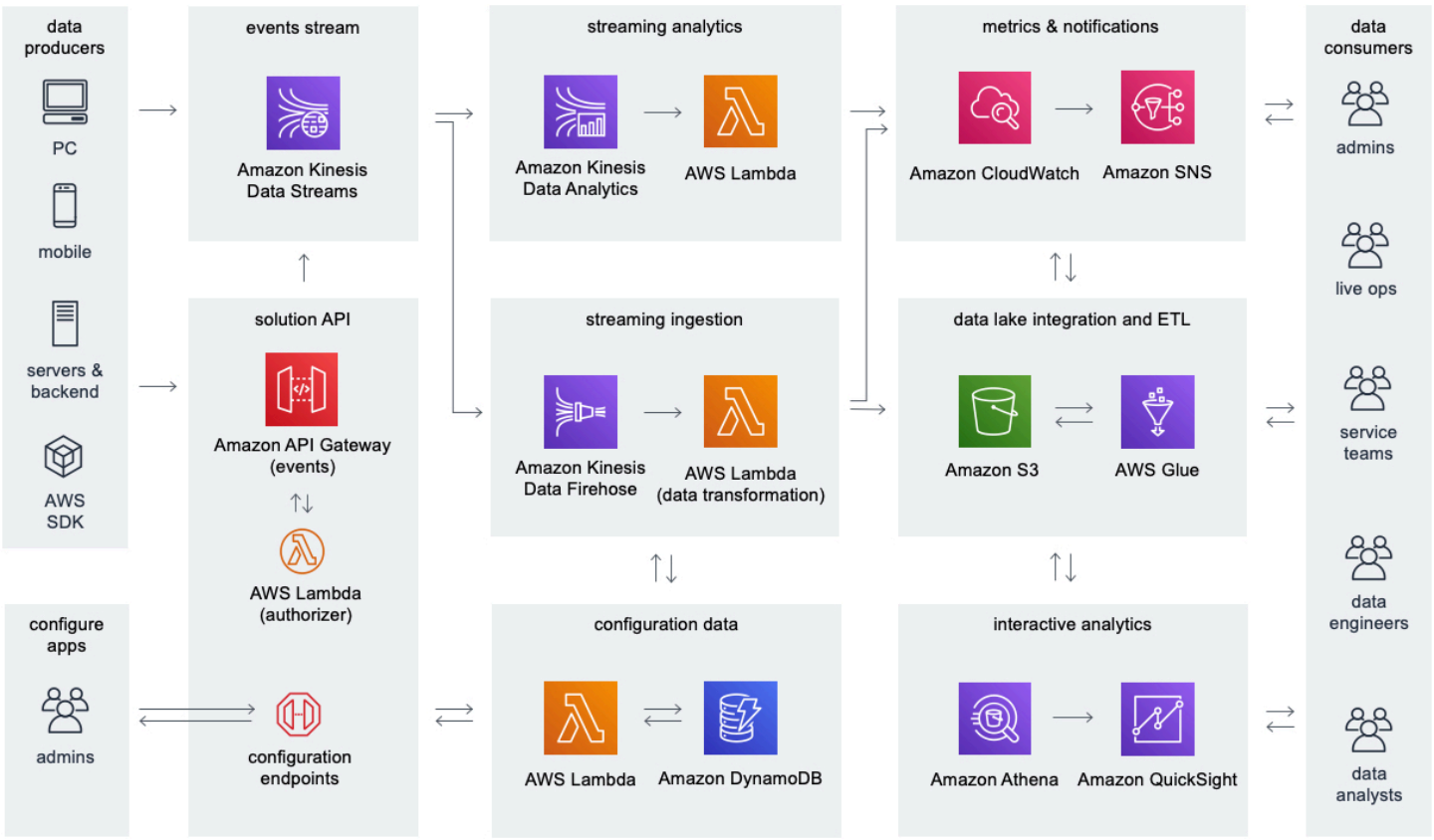
## Pipa analitik game

Pengembang game semakin mencari cara untuk lebih memahami perilaku pemain sehingga mereka dapat meningkatkan pengalaman gameplay untuk mempertahankan dan menumbuhkan basis pemain mereka. Analitik game mewakili infrastruktur teknis dan proses yang diperlukan untuk memahami dan menganalisis data yang dihasilkan dari game dan layanan terkait. Ini biasanya memerlukan penggunaan arsitektur pipeline analitik yang dapat mendukung end-to-end proses ini, seperti implementasi solusi Game Analytics Pipeline.

Arsitektur analitik game memiliki karakteristik sebagai berikut:

- Sumber data mengirim data dalam format umum seperti JSON dan biasanya mencakup server game dan layanan backend game, serta klien game, termasuk PC, perangkat seluler, dan konsol game.

- Pipa analitik game mengotomatiskan seluruh alur kerja menelan dan menyimpan data mentah dan memprosesnya ke dalam format output yang dapat digunakan sehingga dapat dianalisis secara efisien dan hemat biaya oleh konsumen data, seperti pengguna akhir dan aplikasi analitik.
- Pipeline analitik game memberikan dukungan untuk menelan dan memproses volume data real-time yang tinggi untuk diskalakan seiring pertumbuhan game.
- Memberikan dukungan untuk kasus penggunaan pelaporan real-time dan batch. Misalnya, dasbor dan peringatan waktu nyata biasanya digunakan oleh tim operasi langsung untuk memantau infrastruktur game dan perilaku pemain untuk mendeteksi masalah. Tim analis data biasanya mengandalkan pelaporan yang diperlukan dan batch untuk memahami tren dari waktu ke waktu.




Pipa analitik game tanpa server untuk telemetri gameplay

Data game dicerna dari klien game, server game, dan aplikasi lainnya. Data streaming dicerna ke Amazon S3 untuk integrasi data lake dan analitik interaktif. Analisis streaming memproses peristiwa waktu nyata dan menghasilkan metrik. Konsumen data menganalisis data metrik di Amazon CloudWatch dan peristiwa mentah di Amazon S3.

- API Solusi dan data konfigurasi: Gunakan Amazon API Gateway untuk menyediakan REST API untuk mengelola pipeline analitik game Anda dan menyimpan data konfigurasi di Amazon DynamoDB menggunakan fungsi Lambda. Anda dapat membangun portal internal di atas API ini atau antarmuka baris perintah khusus untuk administrasi. REST API juga menyediakan otentikasi server untuk menyerap data gameplay dari sumber data dan meneruskan data telemetri ke Amazon Kinesis Data Streams untuk pemrosesan dan konsumsi waktu nyata ke dalam penyimpanan.
- Aliran acara: Amazon Kinesis Data Streams menangkap data streaming dari game Anda dan memungkinkan pemrosesan data real-time oleh Amazon Data Firehose dan Amazon Managed Service untuk Apache Flink.
- Analisis streaming: Layanan Terkelola untuk Apache Flink menganalisis data peristiwa streaming dari Kinesis Data Streams dan dapat menghasilkan metrik dan peringatan khusus yang dipublikasikan untuk menggunakan fungsi Lambda. CloudWatch
- Metrik dan notifikasi: Gunakan Amazon CloudWatch untuk memantau metrik, log, dan alarm solusi Anda. Gunakan Amazon SNS untuk mengirim notifikasi ke teknisi panggilan dan konsumen data lainnya.
- Konsumsi streaming: Gunakan Firehose untuk mengkonsumsi data streaming Anda dari Kinesis Data Streams dan mengirimkannya ke data lake Anda di Amazon S3 untuk penyimpanan, transformasi, dan integrasi jangka panjang dengan data lain.
- Integrasi data lake dan ETL: Gunakan AWS Glue untuk alur kerja pemrosesan ETL dan untuk mengatur metadata Anda di AWS AWS Glue Data Catalog, yang menyediakan dasar bagi data lake untuk integrasi dengan alat analisis fleksibel.
- Analisis interaktif: Pengguna akhir dapat menggunakan Amazon Athena untuk melakukan kueri interaktif ad-hoc pada kumpulan data yang disimpan di Amazon S3, dan Quick Suite dapat digunakan untuk membuat dasbor.

Lihat Pipeline Game Analytics untuk implementasi referensi otomatis dari pipeline analitik yang dapat digunakan ke akun Anda. AWS CloudFormation

# Definisi

AWS Well-Architected Framework didasarkan pada enam pilar: keunggulan operasional, keamanan, keandalan, efisiensi kinerja, optimalisasi biaya, dan keberlanjutan. AWS menyediakan beberapa komponen inti yang memungkinkan Anda merancang state-of-the-art arsitektur untuk beban kerja game Anda. Pada bagian ini, kami akan menyajikan ikhtisar definisi kunci.

Untuk keperluan paper ini, arsitektur game mencakup infrastruktur teknis backend yang diperlukan untuk membangun dan mengoperasikan game. Beberapa game mungkin tidak memiliki fitur sosial, multipemain, atau online lainnya dan mungkin tidak memerlukan penggunaan aspek-aspek tertentu dari infrastruktur teknis backend yang dijelaskan dalam paper ini. Untuk diskusi mendetail tentang berbagai jenis beban kerja yang sering digunakan untuk mendukung arsitektur game, lihat Skenario.

AWS Cloud Infrastruktur dibangun di sekitar Wilayah dan Zona Ketersediaan.

- A Region adalah lokasi fisik di dunia di mana kami memiliki beberapa Availability Zone.
- Availability Zones terdiri dari satu atau lebih pusat data diskrit, masing-masing dengan daya redundan, jaringan, dan konektivitas, ditempatkan di fasilitas terpisah.

Tergantung pada karakteristik permainan Anda, Anda mungkin ingin menyebarkan komponen tertentu dari arsitektur game Anda ke beberapa Wilayah untuk alasan seperti meningkatkan kinerja untuk pemain, atau untuk memberikan pengalaman yang disesuaikan untuk pemain tergantung pada lokasi mereka.

Ada banyak jenis permainan, dan infrastruktur teknis backend yang diperlukan untuk mendukung permainan berbeda tergantung pada jenis permainan yang sedang dikembangkan. Misalnya, jenis permainan populer mungkin termasuk penembak orang pertama (FPS), permainan peran (RPG), game online multipemain massively (MMOG), battle royales (BR), game olahraga, game puzzle, dan banyak lagi. Ada juga mode interaksi permainan yang berbeda yang memengaruhi arsitektur permainan, seperti permainan berbasis giliran dan simultan, dengan karakteristik kinerja yang berbeda.

Game dikembangkan untuk dimainkan pada satu atau lebih sistem game termasuk desktop, web, seluler, konsol, dan mode interaksi yang lebih baru seperti augmented reality (AR), virtual reality (VR), dan solusi streaming game. Game umumnya mendukung gameplay lintas sistem, yang berarti bahwa pemain dapat menyimpan perkembangan game mereka dan melanjutkan gameplay di sistem lain, serta memulai sesi gameplay dengan pemain di sistem lain.

Monetisasi video game memungkinkan penerbit game menghasilkan pendapatan menggunakan berbagai strategi seperti iklan, pembelian game berbasis digital dan ritel, pembelian konten yang dapat diunduh dalam game (DLC) yang dikenal sebagai transaksi mikro, dan melalui langganan berbayar yang diperlukan untuk memainkan game. Beberapa indikator kinerja utama yang paling umum (KPIs) dalam industri game meliputi:

- Pengguna aktif harian (DAU)
- Pengguna aktif bulanan (MAU)
- Pengguna bersamaan (CCU)
- Durasi sesi
- Biaya per pemasangan (CPI)
- Nilai seumur hidup pemain (LTV)
- Pendapatan rata-rata per pengguna (ARPU)

## Sistem permainan

Video game dikembangkan untuk dimainkan pada sistem game yang menyediakan kontrol input klien, grafik, perangkat lunak klien (dikenal sebagai klien game) dan perangkat keras, dan dalam beberapa kasus fitur eksklusif sistem untuk mendukung gameplay.

Sistem game umumnya dipisahkan ke dalam kategori berikut:

- Konsol: Sistem hiburan yang dibuat khusus yang dirancang untuk bermain game, termasuk contoh populer seperti Sony, PlayStation Microsoft Xbox, dan Nintendo Switch. Konsol menyediakan kemampuan untuk bermain game dengan memasang konten game yang didistribusikan secara fisik atau digital ke perangkat keras konsol yang diproduksi oleh penyedia sistem game. Dalam definisi ini, konsol mungkin genggam atau stasioner dan dimaksudkan untuk digunakan dalam skenario hiburan rumah.
- Komputer pribadi (PC): Game yang dimainkan menggunakan perangkat lunak komputer yang diinstal ke mesin klien yang dapat disesuaikan oleh pemain. Untuk alasan ini, game PC populer di kalangan pemain karena fleksibilitas dan kontrol yang diberikannya.
- Web: Game yang dirancang untuk dimainkan menggunakan browser web dan yang biasanya memberikan manfaat memungkinkan pemain untuk mengakses game terlepas dari sistem operasi mereka.

- Mobile: Game yang dikembangkan untuk dimainkan di ponsel, biasanya sistem operasi smartphone. Game seluler biasanya diunduh dari toko aplikasi digital dan dipasang ke telepon.

Selain sistem yang disebutkan sebelumnya, ada juga sistem yang baru lahir yang masih relatif baru dan berkembang dan memiliki pangsa pasar yang jauh lebih kecil dibandingkan dengan sistem yang lebih dominan. Contoh sistem game dalam kategori ini termasuk AR, VR, dan streaming game, kadang-kadang disebut sebagai cloud gaming.

Streaming game melibatkan rendering gameplay di cloud dan streaming ke thin client, biasanya browser. Game streaming memungkinkan pemain untuk memainkan game yang sepenuhnya di-host dari jarak jauh, biasanya di cloud oleh penyedia layanan streaming game. Dalam streaming game, pemain terhubung ke game berbasis cloud melalui browser atau thin client yang disediakan oleh penyedia layanan cloud gaming (sistem game).

## Server permainan

Server game mewakili salah satu aspek terpenting dari infrastruktur komputasi untuk game Anda. Server game, kadang-kadang disebut sebagai server game khusus, digunakan saat mengembangkan game multipemain atau ketika pemrosesan otoritatif server dari acara gameplay diperlukan. Server game berada di pusat arsitektur game, berfungsi sebagai lokasi di mana logika inti berjalan, yang mencakup mengelola pemain dan status game serta mengelola interaksi antara klien game yang terhubung dan server game. Server game biasanya merupakan salah satu aspek yang paling sensitif terhadap kinerja dari arsitektur game karena bertanggung jawab untuk memproses input dari klien game pemain dan mendistribusikannya dengan benar ke pemain lain yang terhubung secara real-time. Server game yang berkinerja buruk memengaruhi kinerja keseluruhan pengalaman game. Oleh karena itu, Anda harus mengoptimalkan kinerja server game dan menyediakan kapasitas yang cukup, terutama pada peluncuran game atau periode gameplay puncak.

Untuk keperluan dokumen ini, server game atau instance server game mengacu pada komputasi, seperti mesin virtual, yang menghosting satu atau lebih proses server game. Proses server game mewakili satu contoh dari pembuatan server game Anda yang menghosting sesi game, yang merupakan contoh dari game yang sedang berjalan yang dapat dihubungkan oleh pemain melalui sesi pemain. Untuk alasan ini, kita sering merujuk pada proses server game atau sesi game secara bergantian karena hubungan satu lawan satu tersirat antara sesi game dan proses server game yang menghostingnya. Di AWS, ada beberapa opsi untuk komputasi ke server game host, yang menyediakan akses ke kapasitas berbasis cloud yang dapat diskalakan melalui penyediaan sumber daya yang elastis.

Amazon EC2 menyediakan server virtual berbasis cloud, yang dikenal sebagai instance, dengan dukungan untuk beberapa versi Linux dan Windows. Anda dapat membuat instance dan mengelolanya secara langsung seperti server lain atau mesin virtual. Biasanya, beberapa proses server game dikerahkan ke sebuah instance untuk meningkatkan efisiensi dan mengurangi biaya. Amazon EC2 adalah pilihan yang baik untuk server game jika Anda menginginkan kontrol paling besar atas infrastruktur komputasi.

Amazon GameLift menyediakan solusi yang dikelola sepenuhnya untuk hosting server game khusus di cloud serta fitur tambahan seperti perjodohan dengan. GameLift FlexMatch GameLiftmenyediakan lapisan abstraksi di atas Amazon EC2 untuk membuat manajemen server game mudah dan tersedia di sebagian besar Wilayah AWS sehingga Anda dapat meng-host server game yang dekat dengan pemain untuk mengurangi latensi, mencapai ketersediaan tinggi, dan secara signifikan mengurangi biaya dengan menggunakan Instans Spot. Meskipun GameLift dapat diintegrasikan ke dalam backend game yang ada, ini sangat berguna bagi pengembang game yang tidak ingin mengembangkan manajemen server game dan solusi perjodohan mereka sendiri dan lebih memilih solusi yang dikelola oleh AWS dan dapat ditingkatkan seiring pertumbuhan game mereka.

Amazon Elastic Container Service (Amazon ECS) adalah layanan orkestrasi kontainer terkelola penuh yang dapat Anda gunakan untuk menjalankan container berbasis Docker. Anda juga dapat menggunakan Amazon Elastic Kubernetes Service (Amazon EKS) untuk menjalankan container berbasis Docker yang dibuat menggunakan Kubernetes. Menggunakan teknologi kontainer seperti yang disediakan oleh Amazon ECS dan Amazon EKS dapat membantu Anda meningkatkan penggunaan komputasi Anda dengan secara efisien mengemas banyak proses server game atau instance aplikasi game lainnya ke dalam sebuah instance. EC2

Penggunaan kontainer juga dapat meningkatkan produktivitas pengembang dengan menghosting aplikasi menggunakan runtime operasi image Docker yang sama yang digunakan oleh pengembang di mesin lokal mereka selama pengembangan. Anda dapat mengurangi overhead operasional lebih lanjut dengan menggunakan AWS Fargate, yang merupakan solusi komputasi tanpa server untuk menjalankan kontainer dan kompatibel dengan Amazon EKS dan Amazon ECS. Fargate paling cocok untuk kasus penggunaan di mana Anda ingin menjalankan server game dalam wadah tanpa tanggung jawab untuk mengoperasikan instance dasar yang dijalankan kontainer.

Anda dapat menggunakan AWS Outposts untuk menjalankan AWS infrastruktur dan layanan di pusat data atau fasilitas lokal, yang memungkinkan game berjalan di lingkungan lokal dan AWS menggunakan infrastruktur yang sama untuk mendukung strategi adopsi cloud hybrid. AWS Local Zones berfungsi sebagai ekstensi Wilayah AWS yang memungkinkan server game Anda dan beban kerja sensitif latensi lainnya berjalan lebih dekat dengan pemain atau tim pengembangan

Anda. Selain itu, untuk mengurangi latensi jaringan global untuk server game Anda, Anda dapat menggunakan AWS Global Accelerator untuk meningkatkan kinerja lalu lintas pemain ke server game Anda.

AWS Lambda adalah layanan komputasi tanpa server yang menjalankan kode tanpa penyediaan atau pengelolaan server, yang membuatnya berguna untuk kasus penggunaan server game asinkron seperti game berbasis giliran atau yang memiliki persyaratan komputasi ringan, basis kode kecil, dan di mana fungsionalitas gameplay dapat dirancang menggunakan arsitektur microservices stateless. Penting untuk diingat bahwa fungsi Lambda berjalan berdasarkan event-driven, per-request, daripada menjalankan proses server game yang berjalan lama. Lambda menyediakan abstraksi runtime paling banyak dari opsi dalam paper ini karena aplikasi yang mendasarinya sudah tersedia bagi pengembang untuk memilih dari untuk meng-host kode mereka.

Saat memilih pendekatan Anda untuk hosting server game, pertimbangkan berbagai persyaratan termasuk overhead operasional, basis kode lama, persyaratan kinerja, dan skala. EC2instance dan container adalah opsi yang baik untuk basis kode lama, karena memerlukan perubahan terkecil untuk pindah ke cloud, dan Anda dapat menggunakan EC2 instance untuk mendedikasikan sumber daya instance komputasi, sementara kontainer dapat membuat manajemen dan pemanfaatan tinggi lebih mudah dicapai. Fungsi tanpa server menawarkan tingkat abstraksi tertinggi, yang dapat Anda gunakan untuk menentukan kode yang hanya berjalan sebagai respons terhadap peristiwa, yang dapat mengurangi biaya.

## Klien game

Klien game mewakili perangkat lunak dan perangkat keras yang digunakan pemain untuk bermain game. Klien game menyediakan perangkat lunak untuk menerjemahkan input pemain ke dalam pesan yang dikirim ke server untuk diproses, dan bertanggung jawab untuk menangani respons yang masuk dari server dan memberikan output seperti grafik ke pemain. Dalam game multiplayer jaringan real-time, klien game biasanya mempertahankan koneksi jaringan persisten ke server game selama sesi gameplay untuk mengurangi latensi jaringan dan meminimalkan waktu pemrosesan. Namun, klien game juga dapat berinteraksi menggunakan REST dengan server game atau layanan backend.

## Perpesanan

Biasanya ada tiga kategori utama pesan dalam game:

- Pesan keterlibatan pemain yang ditargetkan pada pengguna atau kelompok pengguna tertentu, seperti undangan game atau pemberitahuan push

- Pesan grup antar pemain seperti obrolan dalam game
- ervice-to-servicePesan S, seperti pesan JSON yang digunakan untuk mengintegrasikan dua atau lebih aplikasi

Strategi umum untuk mengirim dan menerima jenis pesan ini adalah dengan menggunakan pola arsitektur pemrosesan penerbit-pelanggan dan asinkron. AWS menyediakan beberapa layanan yang dapat membantu Anda menerapkan pesan dalam game Anda.

- Amazon Simple Notification Service (SNS): Layanan terkelola untuk mengirimkan pesan antara penerbit dan pelanggan menggunakan pola arsitektur. pub/sub Penerbit mengirim pesan menggunakan API ke Amazon SNS, yang mengirimkan pesan secara asinkron ke aplikasi berlangganan dan dapat mengirimkan pemberitahuan push langsung ke klien seluler atau desktop dengan dukungan untuk beberapa layanan pemberitahuan push yang paling banyak digunakan. Amazon SNS dapat digunakan untuk pemberitahuan push ke klien serta kasus penggunaan service-to-service pesan.
- Amazon Simple Queue Service (SQS): Layanan antrian terkelola sepenuhnya yang membuatnya mudah untuk mengintegrasikan server game dan game Anda terlepas dari bahasa pemrograman yang digunakan di masing-masing. Banyak tugas game dapat dipisahkan dan ditangani di latar belakang seperti memperbarui papan peringkat atau nilai waktu bermain dalam database. Pendekatan ini adalah cara yang efektif untuk memisahkan berbagai bagian permainan Anda dan secara mandiri menskalakan fitur yang menghadap pemain dari pemrosesan backend.
- Amazon Managed Streaming for Apache Kafka (MSK): Layanan yang dikelola sepenuhnya yang menyederhanakan pembuatan streaming data dan aplikasi produsen atau konsumen menggunakan Apache Kafka, solusi open-source yang populer. Kafka biasanya digunakan untuk menelan dan memproses data streaming real-time dan dapat digunakan untuk service-to-service pengiriman pesan.
- Amazon ElastiCache (Redis OSS): Menyediakan penyimpanan data dalam memori yang dikelola sepenuhnya yang mencakup dukungan untuk pub/sub fitur populer Redis yang biasa digunakan untuk mengembangkan aplikasi ruang obrolan dan pesan berkinerja tinggi. service-to-service Redis juga mendukung tipe data yang kaya seperti daftar dan set sehingga pengembang dapat menggunakan Redis untuk antrian berkinerja tinggi.
- Amazon Pinpoint: Menyediakan pesan keterlibatan pengguna melalui email, SMS, suara, dan pemberitahuan push. Misalnya, Amazon Pinpoint dapat digunakan untuk mengirimkan pesan keterlibatan pengguna kepada pemain untuk mengundang mereka kembali ke permainan dan

dapat digunakan untuk kasus penggunaan transaksional seperti mendukung token otentikasi multi-faktor, konfirmasi pesanan, dan email pengaturan ulang kata sandi.

## Operasi game langsung (Live Ops)

Operasi game langsung (Live Ops) adalah gaya manajemen dan operasi game yang memperlakukan game sebagai layanan langsung dan terus memberikan fitur baru, pembaruan, promosi, acara dalam game, dan peningkatan pada game yang diluncurkan untuk meningkatkan pengalaman bagi komunitas pemain.

Secara tradisional, game dikirimkan sebagai produk daripada layanan, dan konten dan fitur baru sering dimasukkan ke dalam rilis atau sekuel berikutnya daripada ke dalam produk yang diluncurkan. Dengan pendekatan Live Ops untuk manajemen game, tim operasi game dapat meluncurkan game dan mempertahankan komunitas pemain yang terlibat melalui eksperimen, promosi, acara dalam game, dan inovasi untuk menghibur pemain.

Meskipun pendekatan ini memiliki manfaat membuka strategi keterlibatan pemain baru dan memberikan aliran pendapatan berulang, itu membutuhkan lebih banyak keahlian operasional. Misalnya, untuk menerapkan strategi Live Ops yang sukses, pengembang mungkin perlu berintegrasi dengan layanan cloud atau mengoperasikan infrastruktur teknis backend mereka sendiri. Mereka juga membutuhkan cara yang efektif untuk mengidentifikasi dan menanggapi masalah yang muncul dalam permainan, atau dalam komunitas pemain, yang dapat berdampak negatif pada pengalaman pemain.

# Keunggulan operasional

Pilar keunggulan operasional berfokus pada praktik terbaik untuk menyebarkan dan mengoperasikan game berbasis cloud dalam skala besar. Penting untuk fokus pada keunggulan operasional untuk mempertahankan pengalaman pemain yang positif dan menerapkan langkah-langkah pencegahan untuk mempersiapkan dan memulihkan diri dari masalah yang memengaruhi pengalaman mereka.

Area fokus

- Prinsip desain
- Operasi langsung
- Struktur akun
- Penerapan game
- Pemantauan Kesehatan
- Pengujian beban
- Pengoptimalan dari waktu ke waktu
- Sumber daya

## Prinsip desain

Selain prinsip-prinsip desain dari whitepaper Well-Architected Framework, prinsip-prinsip desain berikut dapat membantu Anda mencapai keunggulan operasional dalam membangun dan mengoperasikan game:

- Tentukan tujuan yang terukur dan dapat dicapai untuk tim operasi game dan beradaptasi seperlunya: Karena sifat permainan yang digerakkan oleh hit, sulit untuk menentukan sebelumnya berapa banyak pemain yang akan memainkan game Anda saat diluncurkan atau harapan apa yang akan dimiliki pemain untuk operasi game Anda yang sedang berlangsung. Penting untuk menetapkan tujuan yang ambisius tetapi dapat dicapai dengan pemangku kepentingan dan merancang pendekatan yang dapat ditingkatkan jika permainan Anda melebihi proyeksi dan diperkecil sementara tim pengembangan game mengoptimalkan pengalaman pemain. Mempersiapkan dan menguji secara memadai sebelumnya untuk memenuhi persyaratan ini dan menyelaraskan pemangku kepentingan bisnis dan teknis Anda pada tujuan target untuk mengoperasikan permainan. Dengan target yang ditentukan, tim game dapat mencapai keseimbangan yang tepat antara biaya dan kinerja selama perencanaan, perancangan, penyediaan, pengujian, penyebaran, dan pengoperasian infrastruktur backend game.

- Gunakan runbook operasional untuk merencanakan kegiatan penskalaan yang terkait dengan peluncuran game dan acara khusus: Tim operasi game harus berkoordinasi dengan pemangku kepentingan bisnis untuk memodelkan proyeksi konkurensi pemain puncak yang diantisipasi untuk acara dan melakukan perencanaan proaktif untuk kapasitas infrastruktur pra-skala sebelumnya. Karena sifat lalu lintas pemain yang berfluktuasi selama acara, perencanaan sebelumnya dan aktivitas pra-penskalaan harus menambah sistem penskalaan otomatis yang ada untuk meningkatkan peluang keberhasilan Anda selama acara dan memverifikasi bahwa Anda memiliki sumber daya yang cukup untuk memberikan pengalaman pemain yang positif. Terapkan praktik rekayasa kinerja untuk mengembangkan dasar sumber daya Anda dan pemahaman berbasis data tentang kapasitas sistem Anda, yang akan membantu memandu aktivitas pra-penskalaan dan konfigurasi penskalaan otomatis. Mengembangkan runbook operasional untuk memberikan konsistensi dalam proses. Perencanaan lanjutan dan respons terhadap permintaan pemain ini sangat penting untuk permainan layanan langsung, yang harus mempertahankan kinerja dan infrastruktur yang andal untuk mendukung basis pemain yang aktif dan terlibat dalam jangka waktu yang diperpanjang.
- Buat model operasi untuk menerima, menyelidiki, dan menanggapi permintaan dukungan pemain: Setelah peluncuran, pantau laporan keluhan dan masalah dengan game. Menerapkan sistem yang tepat untuk berinteraksi dengan pemain dengan cara yang aman dan efektif untuk menyelesaikan masalah pemain secara memadai dan seperti forum komunitas, media sosial, email, sistem tiket, pusat panggilan, atau solusi bot obrolan otomatis. Ini sangat penting untuk permainan layanan langsung, yang membutuhkan komunikasi berkelanjutan dengan basis pemain, responsif terhadap umpan balik pemain untuk memenuhi kebutuhan yang berkembang, dan pemeliharaan komunitas yang terlibat selama masa hidup yang diperpanjang.

# Operasi langsung

GAMEOPS01: Bagaimana Anda menentukan strategi operasi langsung (Live Ops) game Anda?

Kembangkan strategi operasi langsung (Live Ops) untuk game Anda berdasarkan tujuan dan metrik kinerja yang ditentukan, dengan berkonsultasi dengan pemangku kepentingan bisnis.

Praktik terbaik

- GAMEOPS01-BP01 Gunakan tujuan game dan metrik kinerja bisnis untuk mengembangkan strategi operasi langsung Anda

# GAMEOPS01-BP01 Gunakan tujuan game dan metrik kinerja bisnis untuk mengembangkan strategi operasi langsung Anda

Konsultasikan dengan pemangku kepentingan bisnis, seperti produsen game dan mitra penerbitan, untuk menentukan tujuan dan metrik kinerja untuk sebuah game. Ini dapat membantu Anda mengembangkan rencana tentang bagaimana Anda akan mengelola permainan, termasuk menentukan jendela pemeliharaan Anda, jadwal pembaruan perangkat lunak dan infrastruktur, dan keandalan sistem dan tujuan pemulihan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Metrik ini juga dapat membantu Anda menentukan pada tahap siklus hidup game Anda, Anda harus memasukkan steam operasi langsung (Live Ops) untuk memantau kesehatan game, mengumpulkan umpan balik game langsung, dan membangun proses rilis yang efisien dan otomatis. Misalnya, permainan baru mungkin menunggu sampai skala tertentu tercapai, diukur dengan jumlah pemain aktif, pendapatan, atau set metrik lainnya, sebelum menyiapkan tim operasi langsung khusus. Studio pengembangan game yang sudah mapan mungkin sudah memiliki pengalaman operasi langsung, mungkin untuk game mereka yang lain, jadi mereka hanya perlu memasukkan game baru.

Langkah-langkah implementasi

- Anda dapat menentukan target untuk konkurensi pemain (CCU) dan pengguna aktif harian dan bulanan (DAU dan MAU) yang harus dapat didukung oleh infrastruktur game secara efektif, anggaran infrastruktur Anda, target keuangan, dan tujuan kinerja lainnya, seperti frekuensi rilis konten dan fitur untuk meningkatkan keterlibatan pemain. Tujuan dan metrik ini dimasukkan ke dalam keputusan tentang desain game, manajemen rilis, observabilitas, dan dukungan yang diperlukan untuk operasi yang efisien.
- Game Anda mungkin memiliki tujuan untuk merilis pembaruan konten baru setidaknya sekali setiap bulan tanpa downtime selama rilis. Informasi ini membantu Anda menentukan strategi penyebaran rilis dan mengoordinasikan penjadwalan pemeliharaan yang diperlukan yang mungkin memerlukan waktu henti di waktu lain sepanjang bulan dan berkontribusi terhadap ketersediaan SLA Anda.

# Struktur akun

GAMEOPS02: Bagaimana Anda menyusun Akun AWS untuk hosting lingkungan game Anda?

Menerapkan strategi multi-akun untuk mengisolasi lingkungan permainan yang berbeda dan meningkatkan keamanan, efisiensi operasional, dan skalabilitas. Gunakan AWS Organizations untuk mengelola hierarki akun, menerapkan pagar pembatas ke akun, dan menerapkan kebijakan tag dan penandaan pada sumber daya yang diterapkan.

Praktik terbaik

- GAMEOPS02-BP01 Mengadopsi strategi multi-akun untuk mengisolasi berbagai game dan aplikasi ke dalam akun mereka sendiri
- GAMEOPS02-BP02 Mengatur sumber daya infrastruktur menggunakan penandaan sumber daya

## GAMEOPS02-BP01 Mengadopsi strategi multi-akun untuk mengisolasi berbagai game dan aplikasi ke dalam akun mereka sendiri

Merancang struktur akun yang akan memandu penyebaran infrastruktur untuk memenuhi kebutuhan keamanan, isolasi, dan operasional setiap lingkungan. Isolasi lingkungan dengan membatasi akses ke sana dan hanya mengizinkan AWS layanan yang diperlukan untuk digunakan di dalamnya sangat penting, dengan lingkungan produksi terkunci, sementara lingkungan pengembangan dan pengujian lunak untuk memungkinkan eksperimen. Isolasi lebih lanjut dari sub-sistem utama di setiap lingkungan, dan layanan umum yang digunakan oleh beberapa lingkungan untuk di-host dan dikelola sendiri sangat Akun AWS dianjurkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Mengadopsi strategi multi-akun AWS dengan mengisolasi lingkungan yang berbeda (seperti pengembangan, pengujian, pementasan, produksi, dan layanan bersama) ke individu Akun AWS, yang mengurangi ruang lingkup insiden. Pertimbangkan AWS Organizations untuk mengelola hierarki Anda secara terpusat Akun AWS untuk lebih menyederhanakan operasi, serta menentukan dan menerapkan kebijakan tingkat akun dan tingkat unit organisasi (OU-level) secara selektif.

Dengan merancang OU dan Akun AWS struktur yang sesuai yang selaras dengan pengembangan dan kebutuhan alur kerja produksi Anda, Anda dapat mengoptimalkan biaya dan meningkatkan skalabilitas.

- Mengadopsi strategi multi-akun: Mengisolasi lingkungan untuk mengurangi radius insiden dan menyederhanakan operasi.
- Gunakan AWS Organizations: Kelola akun secara hierarkis, terapkan kebijakan, dan aktifkan tata kelola terpusat.
- Rencana untuk Skalabilitas: Merancang struktur akun berbutir halus dan menerapkan langkah-langkah penghematan biaya untuk pertumbuhan di masa depan.

Langkah-langkah implementasi

Sistem game yang digunakan AWS harus menggunakan beberapa akun yang diatur secara logis untuk memberikan isolasi yang tepat, yang mengurangi radius ledakan masalah dan menyederhanakan operasi saat skala infrastruktur game Anda. Akun AWS bahwa infrastruktur game host biasanya dikelompokkan ke dalam lingkungan logis berikut:

- Lingkungan pengembangan game digunakan oleh pengembang untuk mengembangkan perangkat lunak dan sistem untuk game.
- Lingkungan pengujian atau jaminan kualitas (QA) digunakan untuk melakukan pengujian integrasi, QA manual, dan pengujian otomatis lainnya yang harus dilakukan.
- Lingkungan pementasan atau pra-produksi digunakan untuk hosting perangkat lunak yang lengkap sehingga pengujian beban dan asap dapat dilakukan sebelum diluncurkan ke produksi.
- Lingkungan hidup atau produksi digunakan untuk hosting perangkat lunak dan infrastruktur langsung dan melayani lalu lintas produksi dari pemain.
- Layanan bersama atau lingkungan alat menyediakan akses ke sistem umum, perangkat lunak, dan alat yang digunakan oleh banyak tim yang berbeda. Misalnya, repositori kontrol sumber yang dihosting sendiri pusat dan game build farm mungkin di-host di akun layanan bersama.
- Lingkungan keamanan digunakan untuk mengkonsolidasikan log terpusat dan teknologi keamanan yang digunakan oleh tim yang berfokus pada keamanan cloud.

Untuk infrastruktur game aktif AWS, disarankan untuk membuat akun terpisah untuk setiap lingkungan game (pengembangan, pengujian, pementasan, dan produksi), serta akun untuk keamanan, pencatatan, dan layanan bersama pusat.

Biasanya, studio pengembangan game yang lebih kecil yang mengelola sejumlah sumber daya infrastruktur, biasanya beberapa ratus server atau kurang, dapat membuat satu Akun AWS untuk masing-masing lingkungan ini (misalnya, satu akun produksi, satu akun pengembangan, dan satu akun pementasan). Namun, karena infrastruktur game atau ukuran tim Anda tumbuh dari waktu ke waktu, model yang disederhanakan ini mungkin tidak berskala dengan baik.

Saat menyiapkan lingkungan ini, pertimbangkan bahwa banyak AWS layanan berbagi sumber daya dan Service Quotas Tingkat API untuk seluruh akun dalam Wilayah tertentu. Ini harus dipertimbangkan ketika menentukan cara mengatur akun secara logis. Akun AWS hanya mengeluarkan biaya untuk mengkonsumsi layanan yang dikerahkan ke dalamnya. Oleh karena itu, ini menyediakan cara untuk secara efektif mengurangi pertikaian sumber daya dan kuota layanan, terutama ketika game Anda tumbuh dan lebih banyak pengembang memerlukan akses untuk membangun dan mengelola sumber daya.

Berdasarkan pengalaman kami bekerja dengan studio pengembangan game yang lebih besar yang biasanya mengoperasikan ribuan server dengan ratusan pengembang mengakses sumber daya, kami sarankan Anda merancang struktur akun yang lebih halus di mana aplikasi individual yang mendukung game Anda memiliki pengembangan, pengujian, pementasan, dan akun produksi mereka sendiri. Karena sulit dan memakan waktu untuk mendesain ulang strategi AWS multi-akun Anda setelah Anda meluncurkan game karena kompleksitas dalam perencanaan dan migrasi sistem live, pertimbangkan kebutuhan penskalaan masa depan Anda saat menentukan struktur multi-akun yang tepat.

Anda dapat menggunakan AWS Organizationsuntuk mengatur hierarki dan pengelompokan Akun AWS, dan menentukan unit organisasi (OUs) untuk menerapkan kebijakan tingkat OU yang umum kepada mereka melalui kebijakan kontrol layanan (). SCPs AWS Organizations mengelola dan mengatur lingkungan Anda secara terpusat saat Anda tumbuh dan meningkatkan sumber daya Anda. Anda dapat membuat akun baru secara terprogram dan mengalokasikan sumber daya, mengelompokkan akun untuk mengatur alur kerja Anda, menerapkan kebijakan ke akun atau grup untuk tata kelola, dan menyederhanakan penagihan dengan menggunakan metode pembayaran tunggal untuk akun Anda. Selain itu, Organizations terintegrasi dengan layanan lain sehingga Anda dapat menentukan konfigurasi pusat, mekanisme keamanan, persyaratan audit, dan berbagi sumber daya di seluruh akun di organisasi Anda.

AWS Control Towermenyediakan cara mudah untuk mengatur dan mengatur lingkungan multi-akun yang aman, yang disebut landing zone. Control Tower membuat landing zone Anda menggunakan AWS Organizations, membawa manajemen akun dan tata kelola yang berkelanjutan serta menerapkan praktik terbaik berdasarkan AWS pengalaman bekerja dengan ribuan pelanggan

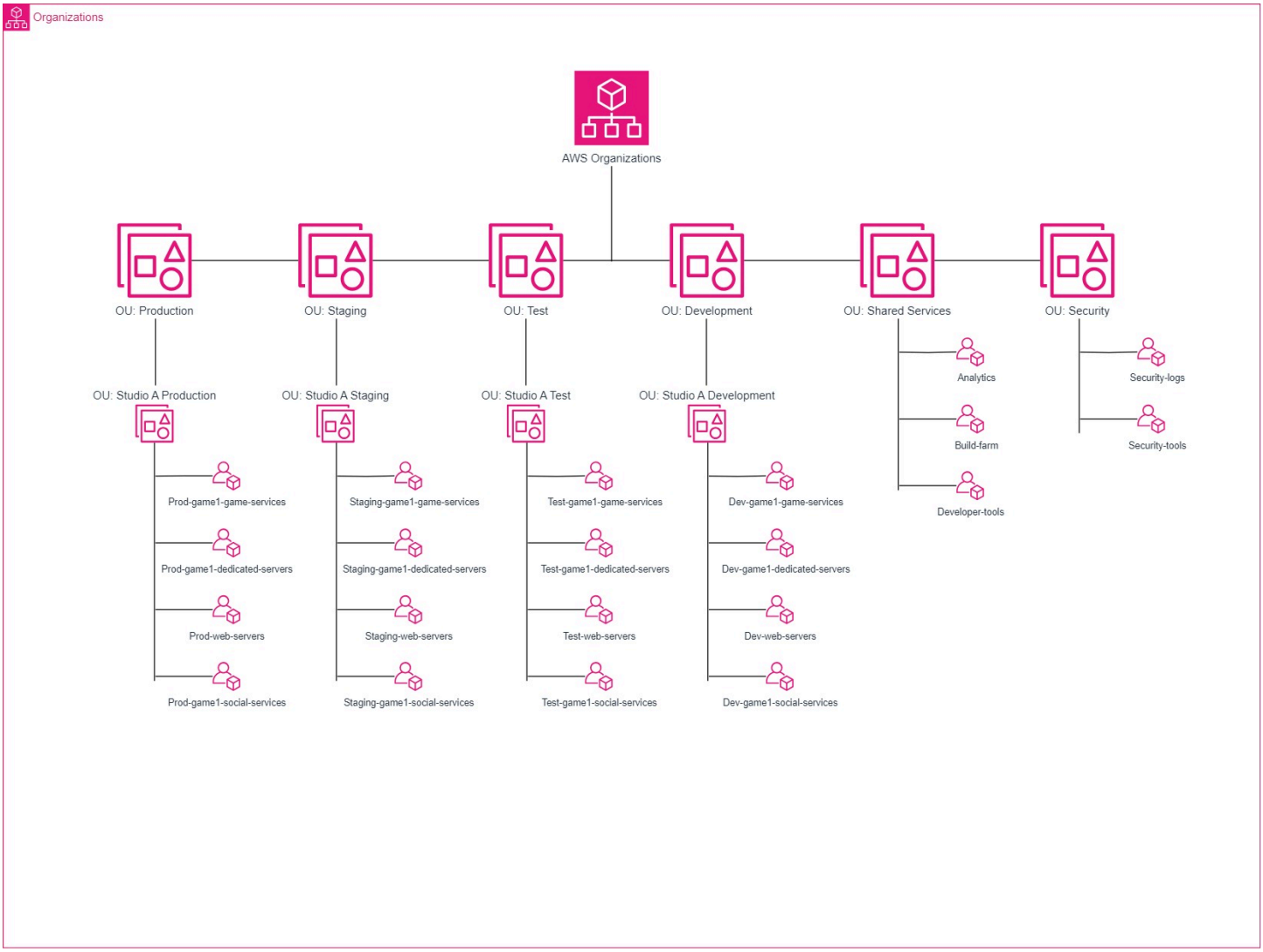
saat mereka pindah ke cloud. AWS Config, AWS Trusted Advisor, dan AWS Security Hub CSPMmerupakan layanan yang memberikan pandangan agregat atau terpusat tentang kebersihan akun Anda.

Isolasi ini membantu Anda mengatur izin dan pagar pembatas khusus atau individual ke setiap lingkungan permainan. Akun produksi harus memiliki pagar pembatas yang diperlukan, pembatasan akses, pemantauan dan peringatan, dan alat keamanan, sementara akun non-produksi mungkin tidak memerlukan tingkat pagar dan izin yang sama. Lingkungan non-produksi dapat diotomatisasi untuk mematikan sumber daya setelah jam kerja dan menghemat biaya. Pemisahan akun pada tingkat granularitas ini membuatnya mudah untuk memantau biaya infrastruktur untuk setiap lingkungan yang mendukung permainan.

Berikut ini adalah contoh struktur multi-akun untuk perusahaan game yang menggunakan AWS Organizations dan unit organisasi (OUs) untuk secara logis mengelompokkan Akun AWS ke dalam lingkungan dan studio yang terpisah. Dalam contoh ini, OUs digunakan untuk mengelompokkan akun berdasarkan lingkungan mereka dan kemudian didasarkan pada studio yang mengoperasikan lingkungan. Ini menunjukkan bagaimana Anda dapat membuat hierarki bersarang untuk memungkinkan aplikasi dan game terpisah untuk digunakan ke akun mereka sendiri dalam lingkungan mereka (digambarkan sebagai OUs), yang dapat berguna jika Anda mengembangkan dan mengoperasikan beberapa game. Lihat dokumentasi dan whitepaper yang disediakan di bagian sumber daya pilar ini untuk mempelajari strategi tambahan yang dapat Anda pertimbangkan untuk mengatur strategi multi-akun Anda.

Berdasarkan pembahasan di atas, contoh diagram di bawah ini mengasumsikan studio game (Organisasi) yang memiliki pipeline pengembangan yang terdiri dari 4 tahap (pengembangan, pengujian, pementasan, dan produksi). Untuk game tertentu (game1), masing-masing lingkungan (OU) memiliki individu Akun AWS untuk layanan game, server game khusus, layanan sosial, dan server web. Sumber daya yang berjalan di masing-masing Akun AWS relevan dengan masing-masing sub-sistem. Biasanya, setiap game individu yang menggunakan pipa pengembangan semacam ini akan mereplikasi ini atau struktur serupa untuknya. Akun AWS

Selain lingkungan game-sentris ini OUs, ada juga layanan bersama OU dan keamanan OU. Ini OUs harus di seluruh organisasi, bukan untuk setiap permainan individu. Dengan begitu game akan menggunakan layanan bersama untuk alat pengembangan dan data dan analitik seperti dalam contoh ini. Kemudian, kirim log aplikasi dan sistem ke Akun AWS pengaturan untuk log di OU keamanan.




Contoh struktur akun untuk lingkungan game

## GAMEOPS02-BP02 Mengatur sumber daya infrastruktur menggunakan penandaan sumber daya

Untuk mengelola dan melacak sumber daya infrastruktur Anda secara efektif AWS, gunakan penandaan dan pengelompokan sumber daya yang tepat untuk mengidentifikasi setiap pemilik sumber daya, proyek, aplikasi, pusat biaya, dan data lainnya. Sumber daya yang ditandai dapat dikelompokkan bersama menggunakan kelompok sumber daya, yang membantu dengan dukungan operasional.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Tentukan kebijakan penandaan. Strategi umum termasuk tag sumber daya untuk mengidentifikasi pemilik sumber daya, seperti nama tim atau nama individu, nama game, aplikasi, atau proyek, nama studio, lingkungan (seperti pengembangan atau produksi), dan peran sumber daya (seperti server database, server web, server game khusus, server aplikasi, atau server cache). Anda dapat menambahkan tag lain untuk membantu kebutuhan bisnis dan TI. AWS Configjuga dapat menerapkan kebijakan penandaan pada waktu pembuatan dan pembaruan sumber daya. Tag dan grup sumber daya tersedia dari Konsol Manajemen AWS, operasi AWS CLI, dan API.

Langkah-langkah implementasi

- Beri tag sumber daya untuk mengidentifikasi pemilik, proyek, aplikasi, pusat biaya, dan data relevan lainnya.
- Menerapkan kebijakan penandaan termasuk tag untuk pemilik, proyek, studio, lingkungan, dan peran sumber daya.
- Gunakan AWS Config untuk menerapkan kebijakan penandaan, dan mengelola tag melalui, Konsol Manajemen AWS CLI, dan API.

## Penerapan game

GAMEOPS03: Bagaimana Anda mengelola penerapan game?

Kelola penerapan game dengan memvalidasi komponen yang digunakan kembali secara menyeluruh, melakukan rekayasa kinerja reguler, dan menerapkan pengujian beban reguler di seluruh siklus hidup pengembangan.

Praktik terbaik

- GAMEOPS03-BP01 Validasi dan uji sistem dan infrastruktur game inti Anda yang ada sebelum menggunakannya kembali di game Anda
- GAMEOPS03-BP02 Melakukan rekayasa kinerja sebelum setiap rilis (atau setidaknya untuk rilis utama)
- GAMEOPS03-BP03 Uji beban lebih awal dan sering
- GAMEOPS03-BP04 Mengadopsi strategi penyebaran yang meminimalkan dampak bagi pemain

- GAMEOPS03-BP05 Infrastruktur pra-skala diperlukan untuk mendukung persyaratan puncak

## GAMEOPS03-BP01 Validasi dan uji sistem dan infrastruktur game inti Anda yang ada sebelum menggunakannya kembali di game Anda

Organizations cenderung menggunakan kembali komponen dan kode sumber yang ada dari game sebelumnya untuk menghemat waktu dan biaya pengembangan. Komponen dan kode lama ini mungkin tidak mengalami tinjauan menyeluruh atau memiliki pengujian integrasi terperinci dan sebagai gantinya bergantung pada kinerja masa lalu mereka.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Sementara penggunaan kembali membantu meningkatkan produktivitas, itu juga dapat memperkenalkan risiko memperkenalkan kembali masalah kinerja dan stabilitas masa lalu ke dalam proyek baru. Oleh karena itu, ketika menggunakan kembali komponen dan kode sumber yang ada dari game sebelumnya, pengujian yang kuat harus diterapkan.

Langkah-langkah implementasi

- Identifikasi kode dan komponen yang digunakan kembali: Katalog kode sumber, pustaka, dan komponen yang digunakan kembali dari game sebelumnya. Bedakan dengan jelas antara kode yang dipelihara secara aktif dan tidak digunakan lagi
- Dokumentasikan perilaku asli dan masalah yang diketahui: Catat karakteristik kinerja asli, keterbatasan fungsional, dan bug yang diketahui atau insiden produksi yang terkait dengan komponen yang digunakan kembali.
- Lakukan tinjauan kode menyeluruh: Lakukan tinjauan teknis terperinci dari komponen yang digunakan kembali, terutama yang memiliki masalah di masa lalu atau didokumentasikan dengan buruk.
- Ganti atau refactor komponen warisan risiko tinggi: Prioritaskan penggantian atau pembaruan komponen lama yang memiliki riwayat masalah atau tidak lagi dapat dipelihara, daripada mengandalkan solusi dalam produksi.
- Melakukan integrasi dan pengujian kompatibilitas: Validasi komponen yang digunakan kembali dalam konteks sistem game baru. Verifikasi bahwa mereka berinteraksi dengan baik dengan modul baru, alat, dan APIs.

# GAMEOPS03-BP02 Melakukan rekayasa kinerja sebelum setiap rilis (atau setidaknya untuk rilis utama)

Rekayasa kinerja adalah proses pemantauan beberapa metrik operasional utama aplikasi untuk menemukan peluang pengoptimalan yang dapat lebih meningkatkan kinerja aplikasi. Ini adalah proses berulang yang dimulai dengan pengujian, diikuti dengan mengoptimalkan kode, dependensinya, proses terkait, sistem operasi host, dan infrastruktur yang mendasarinya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Untuk melakukan analisis kinerja aplikasi yang lebih dalam, integrasikan pemantauan kinerja aplikasi (APM) atau alat debugging dalam kode aplikasi yang dapat mengisolasi masalah dan mengurangi waktu pemecahan masalah dengan melacak perilakunya untuk anomali di seluruh alur aplikasi. Alat APM juga mampu mengidentifikasi metode berkinerja lambat dan operasi eksternal.

AWS X-Raymembantu pengembang dengan aktivitas rekayasa kinerja mereka, seperti mengidentifikasi kemacetan kinerja dan menganalisis dan men-debug kesalahan produksi. Anda dapat menggunakan X-Ray untuk memahami kinerja aplikasi dan layanan yang mendasarinya serta mengidentifikasi serta memecahkan masalah akar masalah kinerja dan kesalahan. Melalui berbagai putaran uji beban, di mana aplikasi dan infrastrukturnya secara bertahap dimuat dengan lalu lintas pemain sintetis, berbagai kemacetan sistem, kesalahan aplikasi, pengecualian, masalah OS, dan masalah lainnya diidentifikasi yang mungkin belum ditemukan selama tes QA lainnya.

Untuk acara penting seperti peluncuran game, rilis konten, promosi, dan acara besar dalam game, gunakan AWS Countdown, yang menyediakan panduan implementasi berdasarkan buku pedoman yang dibuat oleh pakar game untuk memverifikasi kesiapan operasional, mengurangi potensi risiko, dan merencanakan kebutuhan kapasitas.AWS Countdown juga memiliki opsi dukungan premium yang menawarkan dukungan dan opsi yang ditingkatkan seperti insinyur untuk mengoptimalkan infrastruktur Anda.

Langkah-langkah implementasi

- Rekayasa kinerja melibatkan evaluasi dan pemantauan metrik operasional utama untuk memverifikasi bahwa kode, proses, sistem operasi, dan infrastruktur aplikasi Anda berfungsi seperti yang diharapkan. Tinjauan pra-produksi juga membantu menentukan kinerja dasar pada berbagai tingkat penggunaan simulasi.

- Temukan dan lacak metrik utama seperti pemanfaatan, layanan, I/O, proses dan semacamnya dengan menggunakan alat sistem seperti sar, top, vmstat, sysstat, netstat, dan Performance Monitor.
- Lacak kinerja dan perilaku aplikasi Anda menggunakan alat APM seperti AWS X-Ray mengisolasi masalah, mengidentifikasi kemacetan, dan men-debug kesalahan produksi.
- Untuk acara penting seperti peluncuran game, berlangganan AWS Countdown (IEM) untuk panduan arsitektur dan operasional, dukungan operasional sesuai permintaan, dan untuk mengidentifikasi risiko dan merencanakan mitigasi.

## GAMEOPS03-BP03 Uji beban lebih awal dan sering

Pengujian beban adalah proses simulasi lalu lintas dunia nyata pada suatu sistem untuk menilai keandalan dan kinerjanya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Pengujian beban adalah faktor kunci dalam mengembangkan baseline kinerja untuk sumber daya Anda dan memahami kapasitas sistem Anda, yang dapat memandu peramalan keuangan, desain arsitektur, alokasi sumber daya, konfigurasi penskalaan otomatis, dan aktivitas pra-penskalaan pasca-peluncuran. Manfaat tambahan meliputi:

- Infrastruktur yang dioptimalkan: Sumber daya mungkin lebih atau kurang disediakan. Memahami sumber daya yang dibutuhkan akan menghasilkan biaya yang lebih rendah dan infrastruktur yang lebih sedikit untuk dikelola.
- Kesiapan skalabilitas: Mekanisme dan fitur tertentu dapat mendorong pengguna ke dalam game dengan cepat. Mengetahui kapan dan bagaimana menskalakan bisa menjadi perbedaan antara memenuhi permintaan yang meningkat dengan tepat dan kehilangan pemain. Gunakan hasil uji beban untuk menyiapkan runbook dengan ambang batas sistem, titik peringatan, dan titik peringatan kritis pada tingkat penskalaan yang berbeda.
- Kode kualitas yang lebih tinggi: Masalah seperti crosstalk yang berlebihan antar layanan, panggilan basis data yang tidak terbatasi, algoritme yang tidak efisien, kebocoran memori, dan masalah degradasi layanan terkadang lebih mudah diidentifikasi dalam skala besar.
- Validasi perilaku: Menyuntikkan berbagai jenis kegagalan ke dalam pengujian Anda dapat memvalidasi perilaku yang diharapkan sistem atau mengungkap masalah penanganan kesalahan yang perlu diperbaiki.

Idealnya, pengembang harus melakukan pengujian beban di beberapa titik selama proses pengembangan, karena masing-masing dapat menghasilkan manfaat yang berbeda: Sejak awal, mereka memandu keputusan arsitektur dan upaya refactoring sementara itu lebih murah dan mudah untuk membuat perubahan. Pada akhir setiap sprint atau iterasi, mereka memvalidasi kinerja aplikasi dengan fitur dan fungsionalitas terbaru.

Sebelum menerapkan ke produksi, pengujian beban skala besar yang mensimulasikan pola penggunaan dunia nyata yang diharapkan menegaskan kemampuan sistem untuk menangani beban kerja produksi. Setelah penerapan, uji beban berkala memantau kinerja sistem dan mengidentifikasi perubahan atau kemacetan yang mungkin timbul seiring waktu.

Untuk mensimulasikan lalu lintas pemain, Anda memerlukan klien atau bot ringan yang meniru aliran klien game dan bertransaksi dengan backend game untuk mensimulasikan perilaku pemain dunia nyata. Data ini umumnya ditangkap melalui log permainan game dan data yang dihasilkan oleh tes QA yang digerakkan oleh manusia, serta melalui tes alfa atau beta skala terbatas dunia nyata di mana pemain sungguhan diundang untuk memainkan versi permainan akses awal.

Penting untuk mencatat perilaku sistem dalam runbook operasional untuk membantu memecahkan masalah kemungkinan kegagalan di masa depan dan untuk mempertahankan metrik kinerja yang dapat dibandingkan dengan pengujian beban masa depan. Juga disarankan untuk meminta personel QA manusia menguji permainan saat sedang diuji beban karena mereka mungkin menemukan masalah yang gagal diidentifikasi oleh bot dan metrik tidak tercermin.

AWS Fault Injection Service adalah layanan yang dikelola sepenuhnya untuk menjalankan eksperimen injeksi kesalahan yang membuatnya mudah untuk meningkatkan kinerja, observabilitas, dan ketahanan aplikasi. Eksperimen injeksi kesalahan digunakan dalam rekayasa kekacauan, yang merupakan praktik menekankan aplikasi dalam pengujian atau lingkungan produksi dengan menciptakan peristiwa yang mengganggu, seperti peningkatan konsumsi CPU atau memori secara tiba-tiba, mengamati bagaimana sistem merespons, dan menerapkan perbaikan. Eksperimen injeksi kesalahan membantu tim untuk menciptakan kondisi dunia nyata yang diperlukan untuk mengungkap bug tersembunyi, memantau titik buta, dan kemacetan kinerja yang sulit ditemukan dalam sistem terdistribusi.

Langkah-langkah implementasi

- Siapkan lingkungan pengujian beban terdistribusi menggunakan Guidance for Kubernetes-Bases Game Load Testing.

- Kustomisasi dan terapkan kontrol Locust dan pod pekerja di dalam klaster EKS menggunakan file penerapan yang disediakan, memungkinkan pembuatan beban yang dapat diskalakan dan dikelola.
- Rekam perilaku dan metrik sistem selama pengujian beban dalam runbook operasional untuk membantu pemecahan masalah di masa depan dan menetapkan garis dasar kinerja.
- Gunakan eksperimen injeksi kesalahan untuk mensimulasikan gangguan dunia nyata dan mengungkap masalah tersembunyi dalam kinerja sistem, observabilitas, dan ketahanan.

## GAMEOPS03-BP04 Mengadopsi strategi penyebaran yang meminimalkan dampak bagi pemain

Gabungkan strategi penyebaran untuk perangkat lunak dan infrastruktur game Anda yang meminimalkan jumlah waktu henti yang membuat pemain keluar dari permainan Anda. Meskipun jenis pembaruan tertentu mungkin memerlukan pemasangan pembaruan baru ke klien game, rancang game untuk meminimalkan atau menghindari kebutuhan waktu henti selama penerapan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Salah satu langkah paling penting untuk dipertimbangkan ketika mengembangkan strategi penyebaran game adalah menentukan bagaimana infrastruktur game Anda akan dikelola. Kelola infrastruktur game Anda menggunakan alat infrastruktur sebagai kode (IAc) seperti AWS CloudFormationatau Terraform by Hashicorp untuk mengurangi kesalahan manusia selama persiapan lingkungan. Template infrastruktur dapat digunakan dan diuji dalam jaringan pipa otomatis, yang menciptakan konsistensi dalam konfigurasi lingkungan permainan yang berbeda.

Ada beberapa strategi penyebaran yang dapat digunakan untuk permainan:

Substitusi bergulir

Tujuan utama dari substitusi bergulir untuk penyebaran adalah untuk melakukan rilis tanpa mematikan permainan dan tanpa mempengaruhi pemain. Penting bahwa peningkatan atau perubahan yang akan dilakukan kompatibel ke belakang dan akan bekerja berdekatan dengan versi sistem sebelumnya.

Dalam penerapan ini, instance server diganti secara bertahap (diganti atau diluncurkan) dengan instance yang menjalankan versi yang diperbarui. Substitusi bergulir ini dapat dilakukan dengan

beberapa cara berbeda. Misalnya, untuk menerapkan pembaruan bergulir ke armada server game khusus, pendekatan tipikal melibatkan pembuatan grup EC2 instance Auto Scaling baru yang berisi versi build server game baru yang diterapkan ke dalamnya, dan kemudian secara bertahap merutekan pemain ke sesi game yang dihosting di armada server baru ini. Jika ada pembaruan klien game terkait yang diperlukan sebagai prasyarat untuk menggunakan build server game baru, maka Anda harus menyertakan pemeriksaan validasi untuk memverifikasi bahwa hanya pemain yang menginstal pembaruan klien game baru ini yang diarahkan ke sesi game ini.

Armada server (misalnya, grup EC2 Auto Scaling) yang berisi versi build server game lama hanya dihapus dari layanan setelah sesi pemain aktif habis dengan cara yang anggun, biasanya dengan menyiapkan metrik server individual yang memungkinkan tim operasi game untuk mengotomatiskan proses ini. Atau, untuk mengurangi jumlah infrastruktur dan waktu untuk melakukan penyebaran bergulir, pendekatan alternatif dapat dilakukan di mana instance produksi yang ada dihapus dari layanan, diperbarui dengan build server game baru, dan kemudian ditempatkan kembali ke armada produksi. Pendekatan ini mengurangi jumlah infrastruktur yang diperlukan, tetapi juga meningkatkan risiko karena jumlah server game langsung yang tersedia untuk pemain berkurang saat server diganti.

Model ini juga dapat digunakan untuk melakukan penerapan bergulir ke layanan backend seperti database, cache, dan server aplikasi yang tidak menghosting gameplay. Selama layanan ini digunakan dengan cara yang sangat tersedia dengan beberapa instance berkerumun, maka kompleksitas penerapan ke layanan ini harus kurang dari penerapan ke server game khusus.

Penerapan biru/hijau

Tujuan utama blue/green penerapan dalam game adalah untuk meminimalkan waktu henti sementara juga memungkinkan rollback yang aman ke penerapan sebelumnya jika masalah diidentifikasi. Sangat cocok untuk penerapan di mana dua versi backend game kompatibel dan dapat melayani pemain secara bersamaan.

Dalam strategi blue/green penyebaran, dua lingkungan identik (biru dan hijau) diatur. Versi game yang ada diberi label biru, sedangkan versi game baru yang merupakan target penyebaran diberi label hijau. Ketika lingkungan hijau siap untuk migrasi, Anda dapat mengonfigurasi lapisan perutean Anda untuk membalikkan lalu lintas ke lingkungan hijau sambil menjaga lingkungan lama (biru) tersedia jika failback diperlukan. Dalam skenario ini, pembaruan perutean mungkin memerlukan pembaruan layanan perjodohan untuk mengonfigurasinya agar mulai mengirim sesi game ke armada baru, atau dalam kasus layanan backend game, ini bisa memperbarui catatan DNS di Amazon Route 53 untuk layanan Anda atau menggeser bobot penyeimbang beban aplikasi untuk mengirim lalu lintas ke grup target baru Anda.

Salah satu kelemahan dari strategi blue/green penyebaran adalah biaya yang melekat pada lingkungan siaga karena infrastruktur tambahan yang diperlukan saat melakukan penyebaran. Opsi untuk mengurangi biaya infrastruktur tambahan ini adalah dengan mempertimbangkan mengadopsi varian blue/green penyebaran di mana perangkat lunak game baru digunakan ke server yang sama yang sudah digunakan ke dalam produksi. Dalam skenario ini, proses server hijau baru dapat dimulai dengan perangkat lunak baru di samping proses server biru yang ada, dengan pemotongan terjadi antara proses server daripada antara infrastruktur fisik yang terpisah. Pendekatan ini juga dapat mempercepat penyebaran game di sejumlah besar infrastruktur dengan menghilangkan kebutuhan untuk menunggu server baru diluncurkan di cloud. Untuk praktik terbaik tentang pendekatan penerapan ini, lihat Penerapan Biru/Hijau di. AWS

Penyebaran kenari

Penyebaran Canary berguna untuk pengembang game, karena strategi ini dapat diterapkan untuk merilis versi alfa atau beta awal dari sebuah game, atau fitur game seperti mode permainan baru, peta, atau tantangan ke set pemain terbatas atau kecil dalam produksi. Penyebaran seperti itu disebut kenari. Rilis ini mungkin memiliki pelacakan dan pelaporan tambahan, jadi ketika pemain sungguhan memainkan game atau fitur itu, telemetri permainan mereka dikumpulkan dan dianalisis untuk anomali dan masalah.

Untuk fitur baru, para pemain tidak secara konsisten diberitahu tentang hal ini, dan telemetri game adalah sumber utama yang digunakan untuk menentukan apakah pemain mengalami masalah dan rilis harus digulirkan kembali. Pada saat yang sama, jika tidak ada masalah signifikan yang diidentifikasi, fitur tersebut kemudian dapat diluncurkan lebih lanjut ke lebih banyak pemain untuk data tambahan. Jika para pemain diberi tahu, maka mereka dapat diminta untuk memberikan umpan balik reguler tentang pengalaman mereka. Aktivitas pengujian seperti itu idealnya dikoordinasikan oleh tim operasi langsung.

Sebagai strategi, penyebaran kenari juga dapat digunakan untuk rilis standar untuk secara bertahap membuat fitur baru tersedia bagi para pemain. Keuntungan potensial atas blue/green lingkungan standar adalah bahwa lingkungan kedua skala penuh tidak diperlukan. Kapasitas lingkungan baru yang diperkecil menentukan berapa banyak pemain yang akan dionboard ke fitur baru. Sebelum menambahkan lebih banyak pemain, kapasitas harus diskalakan dengan tepat. Bahkan jika blue/green teknik yang disesuaikan ini diharapkan biayanya relatif lebih rendah daripada biru/hijau standar, masih diperkirakan menimbulkan biaya yang mungkin lebih tinggi daripada teknik substitusi bergulir dari penyebaran kenari.

Jalankan hanya satu kenari pada lingkungan produksi, dan fokuskan untuk data dan umpan baliknya. Jika beberapa kenari digunakan, ini mempersulit pemecahan masalah dan mengisolasi masalah dalam produksi dan merusak kualitas kumpulan data dan umpan balik yang dikumpulkan.

Variasi dalam kenari adalah ketika satu atau lebih eksperimen (umumnya pengujian UI) dijalankan melalui penerapan yang ditargetkan, di mana satu set server backend game melayani satu versi fitur dan set berukuran sama lainnya melayani versi lain dari fitur yang sama. Tidak ada infrastruktur tambahan atau khusus yang dibuat untuk ini, dan hanya kantong server backend yang dipilih yang menerima pembaruan ini. Hasil percobaan adalah untuk mengamati bagaimana pemain bereaksi terhadap masing-masing versi fitur yang sama, menentukan apakah ada konsensus suka atau tidak suka secara keseluruhan, dan mengamati apakah ada masalah yang diidentifikasi dengan kegunaan atau fungsinya. Eksperimen strategis semacam itu juga disebut A/B tes, dan keseluruhan proses disebut pengujian A/B. Setelah menyelesaikan eksperimen ini, data pengujian yang diperlukan dikumpulkan sebelum kembali ke versi sistem backend game saat ini di server yang digunakan untuk pengujian.

Penerapan tradisional warisan

Dalam gaya penerapan tradisional, selama jendela pemeliharaan terjadwal permainan dimatikan dan pemain yang terhubung dijatuhkan atau dikeringkan sebelum instance server dalam backend game diperbarui dengan pembuatan kode terbaru. Penyebaran ini memengaruhi pemain setiap kali dilakukan, dan para pemain harus diberi tahu lebih awal dari jadwal. Akibatnya, model ini menyebabkan dampak pemain paling banyak dan harus dihindari bila memungkinkan.

Setelah pembaruan game diterapkan, game dapat diuji asap sebelum membuka game untuk para pemain, yang akan menunggu game dibuka kembali. Hal ini dapat menyebabkan lonjakan lalu lintas ketika pemain mencoba untuk login dan bermain dalam waktu singkat. Oleh karena itu, jika gim ini tidak dirancang untuk menangani lonjakan lalu lintas seperti itu, Anda dapat memilih untuk secara bertahap mengizinkan pemain kembali ke permainan dalam batch.

Atau, Anda dapat memilih untuk menyediakan infrastruktur secara berlebihan untuk mempertahankan lonjakan lalu lintas pembukaan, dan setelah lalu lintas game selesai, sumber daya dapat diperkecil. Jika perlu, lakukan jenis penyebaran ini selama jam-jam off-peak ketika jumlah pemain berada pada titik terendah. Pemeliharaan yang sering dijadwalkan, serta pemeliharaan yang diperpanjang, secara inheren membawa risiko gesekan pemain dan potensi hilangnya pendapatan. Pemain juga mengharapkan perubahan setelah rilis baru dan dapat kehilangan kepercayaan pada permainan setelah kembali setelah periode downtime.

Langkah-langkah implementasi

- Minimalkan waktu henti: Menerapkan strategi penerapan yang mengurangi waktu henti dan menjaga pemain tetap dalam permainan.
- Infrastructure as code (IAc): Gunakan alat seperti AWS CloudFormation atau Terraform untuk mengelola infrastruktur game dan mengurangi kesalahan manusia.
- Strategi penyebaran: Gunakan satu atau kombinasi penggantian bergulir, biru/hijau, dan penyebaran kenari untuk memberikan pembaruan yang lancar dan mengurangi dampak pemain.

## GAMEOPS03-BP05 Infrastruktur pra-skala diperlukan untuk mendukung persyaratan puncak

Skala infrastruktur sebelum acara game skala besar untuk memastikan bahwa Anda dapat menangani peningkatan permintaan pemain yang tiba-tiba.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Selain peluncuran game baru, game langsung biasanya menjalankan acara dalam game, promosi, konten baru, dan rilis musim sebagai contoh cara untuk mempertahankan dan meningkatkan keterlibatan pemain. Kegiatan semacam itu mengalami volume lalu lintas pemain yang tinggi selama acara atau promosi. Bisnis mengharapkan untuk mencapai atau melampaui target yang mereka maksudkan untuk acara tersebut, dan infrastruktur permainan harus mempertahankan dan mendukung mereka melaluinya.

Persiapkan infrastruktur Anda sebelumnya untuk dapat mendukung beban pemain yang diantisipasi yang akan Anda alami selama acara skala besar. Untuk mempersiapkan, tim operasi game harus berkoordinasi dengan pemangku kepentingan dalam penjualan dan pemasaran untuk memperkirakan permintaan yang diproyeksikan yang akan dihasilkan dalam acara mendatang dengan melihat konkurensi pemain sebelumnya, metrik keterlibatan, dan data penjualan. Jika acara tersebut untuk peluncuran game baru, tim operasi game harus bekerja dengan para pemangku kepentingan ini untuk mengidentifikasi proyeksi realistis untuk skala apa yang mereka antisipasi. Meskipun mungkin sulit untuk memprediksi seberapa sukses sebuah game nantinya, penting bagi setiap orang untuk memahami apa harapan untuk sukses sehingga infrastruktur dapat ditingkatkan dan diuji untuk mendukung tujuan tersebut.

Banyak game memilih untuk diluncurkan secara bertahap, dimulai dengan peluncuran lunak dengan membuka game ke sejumlah kecil pemain dan kemudian secara organik menskalakan pemain di setiap tahap, sebelum peluncuran publik penuh. Selama periode peluncuran lunak, pantau, identifikasi, lacak, dan selesaikan masalah sambil menyempurnakan proyeksi Anda untuk peluncuran publik.

Untuk memperkirakan persyaratan infrastruktur dengan benar, kumpulkan data melalui uji beban dan kinerja yang dijalankan terhadap backend game Anda yang berjalan pada produksi atau lingkungan pementasan seperti produksi sebelum peluncuran game. Beberapa putaran pengujian ini harus dijalankan untuk mensimulasikan kondisi permainan yang berbeda dan memvalidasi bahwa backend dapat menahan beban dalam sebagian besar kondisi.

Untuk mencapai ini, pengembang dapat menulis bot gameplay yang melintasi berbagai alur kerja dalam game dan meniru kondisi yang berbeda. Tes ini harus memeriksa lapisan sistem yang berbeda dari backend game sehingga setiap lapisan dan komponen diuji dan detailnya dicatat. Gunakan data yang dikumpulkan dari tes ini untuk menyediakan rencana peluncuran game.

Titik kegagalan tunggal (SPOF) harus diidentifikasi dan dihapus jika memungkinkan dengan membuat aplikasi sangat tersedia dan toleran terhadap kesalahan. Gunakan tes beban untuk mengidentifikasi SPOFs dengan meniru kegagalan pada lapisan hulu dan hilir yang berbeda dan memverifikasi permainan dan perilaku komponen lainnya.

Seiring dengan perkiraan infrastruktur yang diperlukan yang akan disediakan untuk peluncuran game, acara dalam game, atau persiapan promosi, siapkan sistem untuk secara otomatis menskalakan sesuai permintaan. Tentukan, konfigurasikan, dan pantau ambang batas acara penskalaan untuk memungkinkan backend game menskalakan untuk mempertahankan volume lalu lintas pemain yang tinggi. Untuk lalu lintas variabel, pra-penyediaan adalah yang terbaik karena mungkin tidak ada cukup waktu untuk scale-out. Penskalaan manual mungkin diperlukan selama peluncuran game awal yang mendorong permintaan yang lebih tinggi dari yang diantisipasi lebih cepat daripada sistem otomatis yang dapat menskalakan sumber daya.

Pada AWS, organisasi harus meminta Service Quotas yang lebih tinggi untuk layanan yang mereka gunakan di backend game. Service Quotas disiapkan untuk akun untuk melindungi pelanggan dari secara tidak sengaja berdiri atau meningkatkan infrastruktur lebih dari yang dimaksudkan. Ketika game yang berjalan di akun mencapai batas atas kuota layanan yang dikonfigurasi di Wilayah tersebut, layanan akan membatasi permintaan di luar kuota yang disediakan dan ketentuan burst. Throttle dapat menyebabkan kesalahan yang tidak diinginkan atau tidak terduga dan mengganggu pengalaman pemain. Memantau, melacak, dan secara teratur meninjau ambang kuota layanan untuk layanan yang digunakan oleh game dalam produksi untuk menghindari pembatasan. Ketika

penggunaan melewati ambang kuota layanan yang dapat ditoleransi, peningkatan kuota dapat diminta dengan menaikkan Kasus Dukungan dari Pusat Dukungan Konsol, setelah masuk ke akun yang terpengaruh, atau menggunakan Support API.

Untuk acara penting seperti peluncuran game, rilis konten, promosi, dan acara besar dalam game, gunakan AWS Countdown. Countdown memberikan panduan implementasi berdasarkan buku pedoman yang dibangun oleh para ahli Game untuk memberikan kesiapan operasional, mengurangi potensi risiko, dan merencanakan kebutuhan kapasitas.AWS Countdown juga memiliki opsi dukungan premium yang menawarkan dukungan dan opsi yang ditingkatkan seperti insinyur untuk mengoptimalkan infrastruktur Anda.

Jika Anda meluncurkan game yang dihosting di Amazon GameLift, tinjau daftar periksa pra-peluncuran untuk mempersiapkannya.

Langkah-langkah implementasi

- Skalakan infrastruktur ke depan: Siapkan infrastruktur terlebih dahulu untuk acara game skala besar untuk menangani peningkatan permintaan pemain secara tiba-tiba.
- Perkiraan permintaan: Berkoordinasi dengan penjualan dan pemasaran untuk memperkirakan permintaan yang diproyeksikan menggunakan data pemain masa lalu dan proyeksi realistis.
- Pengujian beban dan penghapusan SPOF: Lakukan beberapa putaran uji beban untuk memvalidasi kapasitas backend, mengidentifikasi satu titik kegagalan, dan mengonfigurasi penskalaan otomatis dengan benar.

## Pemantauan Kesehatan

GAMEOPS04: Bagaimana Anda memantau kesehatan game?

Pantau kesehatan game dengan menerapkan instrumentasi komprehensif untuk mendeteksi dan melacak masalah yang berdampak pada pemain, termasuk pencatatan aktivitas sisi klien, pemantauan layanan backend, dan pelaporan kesalahan. Gunakan kombinasi AWS alat seperti Amazon CloudWatch dan AWS X-Ray, serta solusi pihak ketiga, untuk membantu mengidentifikasi dan menyelesaikan masalah dengan cepat.

Praktik terbaik

- GAMESOPS04-BP01 Instrumen permainan untuk mendeteksi dan memantau masalah yang berdampak pada pemain

## GAMESOPS04-BP01 Instrumen permainan untuk mendeteksi dan memantau masalah yang berdampak pada pemain

Selain menanggapi media sosial dan laporan pemain tentang masalah, instrumen permainan Anda dengan solusi pemantauan untuk mendeteksi dan menyelidiki masalah yang berdampak pemain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Tidak ada jumlah pengujian yang dapat mengidentifikasi setiap masalah dalam permainan. Game biasanya diluncurkan dengan masalah yang diketahui yang direncanakan akan diperbaiki secara bertahap dengan rilis game berikutnya. Masalah yang diketahui dan dapat direproduksi mudah diatasi dan diperbaiki. Untuk membantu mengidentifikasi masalah tersebut, klien game harus menerapkan pelacakan aktivitas pemain, pencatatan aplikasi, dan pelaporan di berbagai tempat strategis untuk membantu tim backend mengidentifikasi masalah sisi klien. Kemampuan untuk menemukan masalah seperti itu lebih awal membantu pengembang game memecahkan masalah dan memperbaiki masalah sebelum tersebar luas. Data dan log yang dilaporkan oleh kode pelacakan tidak boleh menyertakan informasi identitas pribadi (PII), dan mereka hanya boleh berisi metadata khusus game yang membantu debugging.

Menerapkan solusi observabilitas untuk mendeteksi dan menanggapi masalah seperti crash game atau bug. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat kenari yang dapat memantau kesehatan layanan permainan backend yang menghadap pemain. Anda dapat menggunakan layanan backend Anda AWS X-Rayuntuk melacak permintaan di seluruh layanan terdistribusi, dan mengirim log dan metrik kustom Anda ke Amazon. CloudWatch

Solusi pihak ketiga, seperti Backtrace.io dan Sentry, adalah solusi populer untuk pelaporan kesalahan dalam game. Solusi pemantauan kinerja aplikasi (APM) dari mitra seperti New Relic, Splunk, Datadog, dan Honeycomb.io juga populer.

Tim operasi langsung permainan dan manajer komunitas juga harus memantau berbagai jejaring sosial dan saluran untuk memeriksa umpan balik pemain, keluhan, dan laporan bug selain saluran dukungan resmi. Tinjau dan coba reproduksi setiap keluhan khusus game, atau kirimkan ke tim QA untuk ditinjau. Jika dapat direproduksi, tingkatkan masalah ke pengembang game untuk pemecahan masalah mereka dan perbaikan sebelum berdampak pada basis pemain yang lebih besar.

Langkah-langkah implementasi

- Menerapkan solusi pemantauan: Gunakan alat pemantauan untuk mendeteksi masalah yang berdampak pemain dan merespons dengan cepat.
- Lacak aktivitas dan log pemain: Klien permainan instrumen untuk mencatat aktivitas pemain dan melaporkan masalah, dan memverifikasi bahwa tidak ada informasi identitas pribadi (PII) yang disertakan.
- Gunakan alat pihak ketiga dan AWS alat: Gunakan alat seperti CloudWatch, X-Ray, dan solusi pihak ketiga untuk pelaporan kesalahan dan pemantauan kinerja, dan pantau media sosial untuk umpan balik pemain dan laporan bug.

# Pengujian beban

GAMEOPS05: Apa yang harus Anda pertimbangkan saat memuat pengujian game?

Saat memuat pengujian game, pertimbangkan tahap pengujian yang sesuai, arsitektur penghasil beban, dan kerangka pengujian untuk mengevaluasi kinerja dan skalabilitas sistem secara efektif. Pilih kombinasi waktu yang tepat (pengembangan awal, sprint, pra-produksi, atau pasca-penyebaran), infrastruktur (, EC2 EKS, Fargate, atau Lambda), dan alat pengujian (, Locust, Grafana K6JMeter, atau Gatling) untuk menyelaraskan dengan karakteristik unik dan tujuan pengembangan game Anda.

Praktik terbaik

- GAMEOPS05-BP01 Pilih kerangka kerja tahap, arsitektur, dan pengujian beban yang tepat untuk memenuhi tujuan Anda

## GAMEOPS05-BP01 Pilih kerangka kerja tahap, arsitektur, dan pengujian beban yang tepat untuk memenuhi tujuan Anda

Pendekatan untuk memuat pengujian permainan dapat sangat bervariasi tergantung pada banyak faktor, termasuk tahap proses pengembangan yang dilakukan, arsitektur sistem penghasil beban itu sendiri, dan pilihan kerangka pengujian beban. Waktu kapan dilakukan, baik pada fase awal, selama sprint iteratif, sebelum penerapan produksi, atau pasca-penyebaran, akan membentuk tujuan dan fokus upaya pengujian. Desain infrastruktur penghasil beban yang berbeda memiliki pro dan kontra

sendiri, dan pemilihan kerangka pengujian beban sangat memengaruhi kemampuan, kemudahan penggunaan, dan integrasi yang tersedia untuk proses pengujian. Dengan menyelaraskan elemen-elemen ini dengan cermat, tim pengembangan dapat menyesuaikan pendekatan pengujian beban dengan karakteristik unik permainan, mengekstrak wawasan kinerja yang paling berharga, dan memberikan pengalaman yang lancar bagi para pemain mereka.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Pengujian beban dalam berbagai tahap pengembangan

Melakukan pengujian beban eksplorasi di awal fase pengembangan dapat memvalidasi arsitektur sistem yang mendasarinya. Ini membantu pengembang untuk membuat keputusan berdasarkan informasi tentang infrastruktur game, desain database, dan topologi jaringan sebelum pekerjaan implementasi ekstensif dilakukan. Tes beban mengidentifikasi risiko dan menciptakan baseline kinerja, berpotensi meminimalkan kebutuhan akan pengerjaan ulang yang mahal dan utang teknis di kemudian hari dalam siklus hidup pengembangan. Mereka juga dapat menumbuhkan pemahaman bersama tentang persyaratan kinerja permainan di antara tim, yang mengarah pada kolaborasi dan pengambilan keputusan yang lebih baik. Pada akhirnya, pengujian beban selama fase awal membangun fondasi yang kuat untuk permainan berkinerja tinggi, terukur, dan tangguh, membantu meningkatkan pengalaman pemain secara keseluruhan.

Pada akhir setiap sprint atau iterasi, pengujian beban dapat mengevaluasi dampak kinerja fitur baru, perbaikan bug, dan perubahan lain yang diperkenalkan dalam siklus terbaru. Pendekatan yang ditargetkan ini memungkinkan tim pengembangan untuk dengan cepat mengidentifikasi regresi atau penurunan kinerja yang diperkenalkan oleh pembaruan terbaru, memungkinkan mereka untuk mengatasi masalah ini sebelum mereka disebarkan lebih jauh ke bawah dan mempertahankan tingkat kualitas dan kinerja yang konsisten.

Sebelum dikerahkan ke produksi, pengujian beban yang kuat membantu tim memvalidasi kemampuan sistem untuk menangani kondisi lalu lintas dan beban dunia nyata yang diantisipasi. Mereka dapat mengungkap hambatan skalabilitas atau kendala sumber daya dalam infrastruktur produksi dan memberikan kesempatan untuk mengoptimalkan kinerja game, menciptakan pengalaman pengguna yang lancar dan responsif sejak hari pertama. Wawasan yang diperoleh dari pengujian beban pra-peluncuran dapat mengurangi risiko hari peluncuran dan menginformasikan perencanaan kapasitas yang sedang berlangsung, yang meletakkan dasar bagi keberlanjutan dan skalabilitas jangka panjang game.

Load testing game yang sudah aktif dalam produksi memungkinkan tim untuk memantau kinerja game dan mengidentifikasi regresi kinerja atau degradasi yang mungkin terjadi dari waktu ke waktu. Hal ini memungkinkan mereka untuk secara proaktif mengatasi masalah sebelum berdampak pada pengalaman pemain dan berdampak negatif pada retensi pengguna. Selain itu, pengujian beban dalam produksi memvalidasi efektivitas upaya optimalisasi kinerja atau penskalaan infrastruktur yang telah diterapkan. Proses ini memberikan pengalaman bermain game berkualitas tinggi, responsif, dan terukur bagi para pemain bahkan saat game berkembang dan matang.

Arsitektur penghasil beban

Desain arsitektur penghasil beban untuk pengujian beban game dapat mengambil berbagai bentuk, masing-masing dengan serangkaian keunggulan dan pertimbangannya sendiri.

Pada tingkat yang paling dasar, EC2 instans Amazon yang dikelola sendiri dapat disediakan dan dikonfigurasi untuk bertindak sebagai generator beban. Dengan pendekatan node kontrol dan node pekerja, Anda dapat mengatur beberapa instance penghasil beban, masing-masing menjalankan skrip pengujian mereka sendiri dan secara keseluruhan dikelola oleh satu instance kontrol. Arsitektur dapat meningkatkan dan menghasilkan lebih banyak beban tanpa meningkatkan kompleksitas dengan memutar node pekerja tambahan, tetapi pendekatan langsung ini mengharuskan tim untuk menangani penyediaan, konfigurasi, dan pengelolaan infrastruktur yang mendasarinya.

Untuk pendekatan yang lebih skalabel dan terorkestrasi, Anda dapat menggunakan klaster Amazon EKS Kubernetes untuk mengelola dan mendistribusikan beban kerja pengujian beban di seluruh armada agen beban berbasis kontainer. Fitur penskalaan otomatis Kubernetes dapat digunakan untuk menangani penskalaan pod yang menghasilkan beban, sementara tim sendiri mengonfigurasi dan mengelola EC2 instans yang mendasarinya di cluster yang menghosting pod.

Atau, sifat tanpa server AWS Fargatedapat mempercepat dan menyederhanakan pengaturan pengujian beban dengan mengabstraksi manajemen infrastruktur sambil tetap memberikan skalabilitas dan fleksibilitas yang diperlukan. Untuk solusi hybrid di mana klaster Kubernetes lokal yang menghasilkan beban sudah ada tetapi kapasitas tambahan mungkin diperlukan, EKS Anywhere dapat mengelola kedua klaster sebagai salah satunya. Konsol Manajemen AWS

Anda juga dapat menggunakan AWS Lambdafungsi tergantung pada kebutuhan dan tujuan Anda. Fungsi Lambda relatif mudah diatur dan diskalakan tanpa perlu menyediakan dan mengelola sumber daya tambahan. Mereka juga memungkinkan pembuatan skenario pengujian yang lebih kompleks dan dinamis karena integrasi mendalam dengan AWS layanan lain. Namun, fungsi Lambda memang memiliki batasan pada fungsi dan runtime bersamaan (15 menit), yang dapat membatasi skala dan panjang pengujian beban yang dapat dicapai. Latensi start dingin juga dapat memengaruhi

keakuratan hasil, dan keterbatasan sumber daya Lambda mungkin tidak cocok untuk beban kerja pengujian beban yang sangat menuntut.

Studio yang ingin menggunakan solusi pra-bangun dapat menggunakan Pengujian Beban Terdistribusi. AWS Solusi ini menggunakan Amazon ECS AWS Fargate untuk menyebarkan kontainer yang dapat menjalankan simulasi puluhan ribu pengguna yang terhubung. Anda dapat menggunakan ini untuk memulai infrastruktur pengujian beban Anda dengan cepat dalam mode IAC menggunakan AWS CloudFormation.

Memuat kerangka pengujian

Tidak ada dua kerangka pengujian beban yang dibangun sama. Beberapa memiliki antarmuka grafis intuitif untuk pembuatan pengujian, sementara yang lain sepenuhnya berbasis baris perintah. Satu alat mungkin fleksibel dan berkinerja tetapi membutuhkan waktu dan upaya untuk mengkonfigurasi dan mengelola, dan yang lain mungkin tanpa server tetapi terbatas dalam pengujian yang dapat dibuat dan dijalankan. Beberapa menikmati komunitas besar dan banyak tutorial sementara tidak terbukti di lapangan, sangat kontras dengan yang lain yang mungkin diuji dalam pertempuran dalam produksi tetapi tidak memiliki dukungan atau dokumentasi komunitas. Pilih kerangka kerja yang memberikan keseimbangan yang tepat untuk Anda dan tim Anda. Beberapa opsi populer adalah:

- Apache JMeter: Kerangka pengujian beban open-source berbasis Java yang populer karena rangkaian fiturnya yang kuat dan kemudahan penggunaan. Kemampuannya untuk mensimulasikan skenario pengguna yang kompleks, berbagai protokol yang didukung, pelaporan komprehensif, dan rekam jejak yang terbukti membuat pilihan JMeter yang andal untuk pengujian beban.
- Locust: Kerangka pengujian beban terdistribusi modern yang dibangun di atas arsitektur berbasis peristiwa, membuatnya berkinerja sementara hemat sumber daya. Pengujian ditulis dengan Python, memungkinkan skenario pengujian fleksibel yang memanfaatkan ribuan pustaka pihak ketiga yang kuat, sambil tetap ramah dan mudah dibaca.
- Grafana K6: Kerangka pengujian beban yang kuat yang menggabungkan kemudahan penggunaan dengan kemampuan canggih. Dukungannya untuk pembuatan beban terdistribusi, skrip fleksibel, dan integrasi tanpa batas dengan Grafana untuk visualisasi data menjadikan Grafana K6 pilihan yang menarik.
- Gatling: Kerangka pengujian beban sumber terbuka yang dikenal karena kinerja dan skalabilitasnya. Bahasa khusus domain (DSL) berbasis SCALA memungkinkan pengembang untuk membuat skrip pengujian beban yang ringkas dan dapat dipelihara, dan kemampuan pelaporan dan analisisnya yang kuat memberikan wawasan terperinci tentang sistem yang diuji.

Langkah-langkah implementasi

- Tahap pengujian beban: Melakukan pengujian beban pada berbagai tahap pengembangan (pengembangan awal, sprint, pra-produksi, dan pasca-penerapan) untuk memvalidasi kinerja sistem dan mengidentifikasi masalah.
- Arsitektur penghasil beban: Pilih arsitektur penghasil beban yang sesuai (, EC2 EKS, Fargate, atau Lambda) berdasarkan kebutuhan skalabilitas, preferensi manajemen, dan persyaratan pengujian khusus.
- Kerangka kerja pengujian beban: Pilih kerangka pengujian beban (seperti JMeter, Locust, Grafana K6, atau Gatling) yang menyeimbangkan kemudahan penggunaan, kinerja, fleksibilitas, dan dukungan komunitas agar sesuai dengan kebutuhan tim Anda.

# Pengoptimalan dari waktu ke waktu

GAMEOPS06: Bagaimana Anda mengoptimalkan game Anda dari waktu ke waktu?

Optimalkan permainan Anda dari waktu ke waktu dengan memantau metrik utama dan data telemetri untuk mengidentifikasi tren pemain, kinerja sistem, dan area untuk perbaikan. Terus perbarui desain game, infrastruktur, dan pendekatan pengujian beban Anda berdasarkan wawasan ini, sambil beradaptasi dengan teknologi dan kerangka kerja baru untuk memberikan kinerja dan pengalaman pemain yang optimal saat game Anda berkembang.

Praktik terbaik

- GAMEOPS06-BP01 Memantau metrik permainan utama untuk mengidentifikasi tren dan pola pemain, dan menggunakan informasi untuk meningkatkan permainan
- GAMEOPS06-BP02 Perbarui dan sesuaikan pendekatan pengujian beban saat game berubah

## GAMEOPS06-BP01 Memantau metrik permainan utama untuk mengidentifikasi tren dan pola pemain, dan menggunakan informasi untuk meningkatkan permainan

Selain penggunaan sistem klien game, penggunaan aplikasi, pengecualian, dan data crash, tangkap data telemetri game yang dikirim ke sistem backend game. Data ini harus mewakili aktivitas pemain

sehingga Anda dapat memahami bagaimana pemain berinteraksi dengan berbagai fitur dalam permainan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Bergantung pada implementasinya, klien game dapat mengumpulkan data telemetri di fitur atau lokasi game yang telah ditentukan sebelumnya di dunia game. Data dikirim ke layanan konsumsi backend untuk diproses. Jika layanan backend tidak dapat dijangkau, klien dapat menyimpan data secara lokal di perangkat lokal hingga layanan backend tersedia lagi. Perancang game menggunakan data telemetri ini untuk meninjau bagaimana pemain memainkan game, dan jika ada anomali dalam game.

Misalnya, gerakan pemain dan interaksi dengan item dalam peta dapat diekstraksi dari data telemetri dan diplot sebagai peta panas aktivitas dalam game oleh pemain selama jangka waktu tertentu. Data tersebut membantu desainer game mengidentifikasi kebutuhan untuk menyeimbangkan berbagai elemen dalam game, seperti kekuatan senjata, kekuatan karakter dalam game, atau kompleksitas peta. Data telemetri mentah umumnya disimpan dan kemudian diproses untuk mengekstrak analitik yang dapat divisualisasikan oleh analis.

Implementasi AnalyticsPipeline solusi Game membantu pengembang game meluncurkan pipeline data tanpa server yang dapat diskalakan untuk menyerap, menyimpan, dan menganalisis data telemetri yang dihasilkan dari game dan layanan. Solusi ini mendukung streaming konsumsi data, memungkinkan pengguna untuk mendapatkan wawasan dari game mereka dan aplikasi lain dalam beberapa menit.

Untuk konsumsi data telemetri game kustom, penyimpanan, pemrosesan, dan analitik, AWS juga menawarkan sejumlah layanan khusus untuk pemrosesan data besar dan Analytics.

Langkah-langkah implementasi

- Tangkap data telemetri game: Kumpulkan data tentang aktivitas pemain, penggunaan sistem, pengecualian, dan crash untuk memahami interaksi pemain dan mengidentifikasi masalah.
- Menerapkan koleksi telemetri: Gunakan fitur atau lokasi permainan yang telah ditentukan sebelumnya untuk mengumpulkan data telemetri dan mengirimkannya ke layanan backend, simpan secara lokal jika backend tidak dapat dijangkau.
- Gunakan solusi AWS analitik: Gunakan AWS layanan seperti Game Analytics Pipeline untuk konsumsi, penyimpanan, dan analisis data yang dapat diskalakan, serta layanan pemrosesan dan analitik data besar khusus.

# GAMEOPS06-BP02 Perbarui dan sesuaikan pendekatan pengujian beban saat game berubah

Mengoptimalkan pendekatan pengujian beban adalah proses berkelanjutan yang harus berkembang seiring dengan siklus pengembangan game. Saat game tumbuh dalam kompleksitas, basis pengguna, dan kumpulan fitur, strategi pengujian beban harus beradaptasi untuk memverifikasi bahwa game tersebut secara akurat mensimulasikan kondisi dunia nyata dan memberikan wawasan yang dapat ditindaklanjuti.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Pertimbangkan hal berikut:

Skenario pengujian yang hilang atau ketinggalan zaman

Karena fungsionalitas baru ditambahkan ke game selama proses pengembangan, buat dan jalankan skenario pengujian beban baru untuk memvalidasi kinerja dan skalabilitas fitur baru. Demikian pula, fitur dan fungsionalitas sering difaktorkan ulang untuk meningkatkan kinerja, mengatasi umpan balik pemain, atau menyelaraskan dengan tujuan desain baru, yang membutuhkan skenario pengujian untuk terus diperbarui untuk mengimbangi perubahan dan benar-benar menguji dan mencerminkan keadaan sistem.

Kerangka pengujian beban baru

Pengembang mungkin perlu mengubah kerangka pengujian beban karena berbagai alasan:

- Kerangka kerja awal mungkin tidak lagi dapat mensimulasikan beban pengguna secara memadai atau memberikan tingkat wawasan yang diperlukan tentang kinerja sistem
- Fitur permainan baru mungkin memerlukan dukungan pengujian beban untuk protokol baru, APIs, atau titik integrasi
- Pengembang mungkin menginginkan fitur yang lebih canggih karena mereka menjadi lebih nyaman dengan proses pengujian beban
- Preferensi untuk kerangka kerja yang lebih selaras dengan keahlian teknis tim, bahasa pemrograman, atau rantai alat yang ada

Dengan mengevaluasi dan beradaptasi secara hati-hati dari waktu ke waktu, pengembang dapat menyelaraskan proses pengujian beban dengan persyaratan permainan yang berubah dan terus

memberikan wawasan yang diperlukan untuk mengoptimalkan dan meningkatkan pengalaman pengguna secara keseluruhan.

Mengoptimalkan biaya

Kemudahan dan kenyamanan menggunakan AWS layanan yang dikelola dapat sangat bermanfaat, terutama pada tahap awal pengembangan. Layanan ini mengabstraksikan manajemen infrastruktur yang mendasarinya, memungkinkan tim untuk dengan cepat menyiapkan solusi mereka dan hanya fokus pada pembuatan skenario pengujian beban dan menganalisis hasilnya. Namun, menggunakan layanan terkelola seringkali dapat menghasilkan biaya yang lebih tinggi karena nilai tambahan dan kenyamanan yang mereka berikan, seperti penyediaan, konfigurasi, dan pemeliharaan infrastruktur, serta menyediakan kemampuan ketersediaan, penskalaan, dan pemantauan yang tinggi.

Ketika tim matang dan tumbuh lebih nyaman dan percaya diri dengan proses pengujian beban mereka, mungkin ada saatnya ketika mengelola sendiri infrastruktur dapat memberikan pengoptimalan tambahan dan penghematan biaya. Meskipun pendekatan langsung ini meningkatkan overhead operasional, memiliki kontrol langsung atas sumber daya komputasi, konfigurasi, perilaku penskalaan, dan pemanfaatan sumber daya dapat membuka peluang baru untuk menyempurnakan dan mengurangi biaya. Misalnya, mungkin masuk akal bagi tim untuk memulai perjalanan pengujian pemuatan mereka dengan arsitektur AWS Fargate tanpa server, lalu pindah ke mengelola sendiri node yang mendasarinya di cluster Amazon EKS nanti.

Langkah-langkah implementasi

- Perbarui skenario pengujian: Terus membuat dan memperbarui skenario uji beban untuk memvalidasi fitur baru dan fungsionalitas yang difaktorkan ulang, dan memverifikasi bahwa skenario tersebut mencerminkan keadaan game saat ini.
- Mengevaluasi kerangka pengujian beban: Beradaptasi dengan kerangka kerja baru sesuai kebutuhan untuk mensimulasikan beban pengguna, mendukung protokol baru, dan menyelaraskan dengan keahlian dan rantai alat tim.
- Optimalkan biaya: Mulailah dengan AWS layanan terkelola untuk kemudahan dan kenyamanan, kemudian pertimbangkan infrastruktur pengelolaan mandiri untuk penghematan biaya karena tim semakin nyaman dengan proses pengujian beban.

# Sumber daya

Lihat sumber daya berikut untuk mempelajari lebih lanjut tentang praktik terbaik kami terkait keunggulan operasional.

# Dokumentasi dan blog

- Praktik terbaik arsitektur untuk Game Tech
- Mengelola Studio Game Anda di AWS pt.1
- Mengelola Studio Game Anda di AWS pt. 2
- Mengelola Studio Game Anda di AWS pt. 3
- Membangun AWS lingkungan praktik terbaik Anda
- strategi multi-akun untuk landing zone Control Tower
- Pipa Analitik Game
- Maksimalkan Wawasan Data Game Anda dengan Pipeline Analytics Game
- Memanfaatkan AWS Glue dan Amazon Redshift Spectrum untuk Wawasan Pemain
- Cara mengatur CI/CD pipa
- Bagaimana Good Job Games Berakselerasi 43% dengan AWS Build Pipeline
- Menerapkan Build Pipeline untuk Unity Mobile Apps
- CI/CD Blog lain yang relevan
- Game DevOps menjadi mudah dengan blog Game-Server CD Pipeline
- Harmony Games Menyebarkan Backend Game yang Sepenuhnya Kustom Menggunakan () AWS Cloud Development Kit (AWS CDK)AWS CDK
- GameLiftBersiaplah untuk peluncuran
- Hosting server game hybrid dengan Amazon Gamelift Anywhere
- Percepat pengembangan server game dengan Amazon Gamelift Anywhere dan Amazon Gamelift Agent
- Cara meng-host game Unreal Engine Anda dengan harga di bawah $1 per pemain dengan Amazon Gamelift
- Panduan Solusi Baru untuk membangun backend game lintas platform yang dapat diskalakan AWS
- Muat pengujian mesin game backend Pragma ke 1 juta pengguna bersamaan di AWS
- Bagaimana pemuatan Code Wizards menguji Nakama Heroic Lab ke dua juta pemain bersamaan AWS
- Praktik terbaik dengan Unit Organisasi
- AWS X-Ray
- AWS Hitung mundur

- AWS untuk Hub Solusi Game

## Solusi mitra

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

## Whitepaper

- Mengatur Lingkungan Anda menggunakan beberapa akun
- Pengantar Pola Pengembangan Game yang Dapat Diskalakan AWS

## Video

- YouTubeseri: Membangun Game di AWS
- AWS untuk Game: Boss LEVEL Podcast
- Re:Invent 2023: Menskalakan AWS untuk 10 juta pengguna pertama
- Re:Invent 2022: Bagaimana Riot Games memproses 20TB analitik setiap hari di AWS
- Re:Invent 2022: How AWS and Riot Games membangun mesin pelaporan tata kelola
- Re: Invent 2023: Menerapkan pola desain terdistribusi pada AWS
- Re:Invent 2023: Menskalakan game multipemain menjadi jutaan dengan Mortal Kombat 1
- Re: Invent 2022: Evolusi rekayasa kekacauan di Netflix
- Re:Invent 2023: Praktik terbaik untuk tata kelola cloud
- Re:Invent 2023: Praktik terbaik untuk membuat arsitektur Multi-wilayah di AWS

## Materi pelatihan

- Kurikulum — Memulai AWS untuk Game Bagian 1

# Keamanan

Pilar keamanan mencakup kemampuan untuk melindungi informasi, sistem, dan aset sambil memberikan nilai bisnis melalui penilaian risiko dan mitigasi. Karena visibilitas global dan sejumlah besar pemain, game adalah target yang diinginkan untuk eksploitasi, peretas, dan lainnya yang mencari cara untuk mengeksploitasi dan menyalahgunakan sistem. Hal ini sering dapat menghasilkan pengalaman pemain yang mengecewakan dan peningkatan biaya bagi pengembang game jika tidak ada fondasi keamanan yang kuat yang ditetapkan.

Seperti yang dijelaskan dalam Model Tanggung Jawab Bersama, penting untuk memahami aspek keamanan mana yang menjadi tanggung jawab AWS dan aspek mana yang menjadi tanggung jawab pelanggan sehingga Anda siap untuk mempertahankan postur keamanan yang kuat. Pilar ini memberikan panduan keamanan cloud praktik terbaik untuk Anda pertimbangkan saat mengembangkan dan mengoperasikan game di cloud.

Sebelum Anda merancang suatu sistem, Anda harus menetapkan serangkaian praktik terbaik keamanan yang mencakup kontrol akses. Selain itu, Anda harus dapat mengidentifikasi insiden keamanan dan melindungi sistem dan layanan Anda sambil menjaga kerahasiaan dan integritas data melalui perlindungan data. Anda harus memiliki proses yang terdefinisi dengan baik dan dapat dipraktikkan untuk merespons insiden keamanan. Alat dan teknik ini penting karena mendukung tujuan bisnis seperti mencegah kerugian finansial atau mematuhi kewajiban peraturan.

Contoh pelanggan

AnyCompany Game adalah studio game fiksi yang sedang dalam proses meningkatkan postur keamanan mereka. Keamanan dapat mudah dipahami ketika ada penjelasan tentang aplikasi langsungnya. AnyCompany Game digunakan di bagian ini untuk mengontekstualisasikan praktik terbaik keamanan yang dijelaskan dalam pilar

Area fokus

- Prinsip desain
- Fondasi keamanan
- Keamanan yang sedang berlangsung
- Manajemen identitas dan akses
- Kontrol akses
- Deteksi
- Perlindungan infrastruktur

- Respons insiden
- Keamanan aplikasi
- Mengotomatiskan keamanan
- Pemodelan ancaman
- Sumber daya

## Prinsip desain

Selain prinsip-prinsip desain dari pilar keamanan whitepaper Well-Architected Framework, prinsip desain berikut dapat memperkuat keamanan beban kerja game Anda di cloud:

- Memantau dan memoderasi perilaku penggunaan pemain: Tangkap dan analisis data penggunaan untuk memahami bagaimana pemain berinteraksi dengan permainan dan fitur sosial Anda. Dengan menganalisis data ini, Anda dapat mendeteksi dan menanggapi perilaku kasar dan tidak pantas yang dapat menurunkan pengalaman pemain.

## Fondasi keamanan

GAMESEC01: Bagaimana Anda menerapkan dasar-dasar keamanan untuk pengembangan game?

Studio game memerlukan pendekatan keamanan unik yang melindungi lingkungan pengembangan dan layanan pemain langsung. Strategi AWS keamanan yang kuat untuk studio game membutuhkan tiga komponen yang saling berhubungan: struktur multi-akun, otentikasi yang kuat, dan strategi otorisasi yang jelas menggunakan kebijakan IAM. AWS Struktur multi-akun memungkinkan studio untuk memisahkan berbagai proyek game, tahapan pengembangan, dan lingkungan alat. Ini memberi studio kontrol yang lebih terperinci untuk hal-hal seperti akses ke lingkungan atau layanan tertentu. Mengaktifkan otentikasi yang kuat memverifikasi anggota tim dapat mengakses sumber daya pengembangan dengan aman baik bekerja di studio atau jarak jauh, sambil mempertahankan kontrol ketat atas kode sumber, pembuatan game, dan alat berpemilik. Studio juga harus memiliki strategi otorisasi yang jelas untuk memberikan izin menggunakan prinsip hak istimewa terkecil dengan izin dan peran IAM. Gunakan peran IAM untuk menetapkan izin antara peran tim pengembangan yang berbeda, seperti memberi tim pengembang akses ke AWS layanan tingkat

rendah sambil membatasi artis dan desainer untuk manajemen aset tertentu dan membangun sistem. Pendekatan khusus ini memverifikasi studio game dapat melindungi kekayaan intelektual mereka, mempertahankan alur kerja pengembangan yang efisien, dan meningkatkan skala tim mereka dengan aman sambil memberi pengembang akses yang tepat untuk mengulangi proyek mereka dengan cepat.

Praktik terbaik

- GAMESEC01-BP01 Gunakan peran dan akses federasi, bukan pengguna root akun, untuk melakukan tindakan di lingkungan Anda AWS
- GAMESEC01-BP02 Gunakan AWS Control Tower untuk mengatur lingkungan multi-akun dengan cepat AWS
- GAMESEC01-BP03 Gunakan kebijakan peran hak istimewa terkecil yang disesuaikan dengan fungsi pekerjaan tertentu
- GAMESEC01-BP04 Gunakan peran dan kebijakan akses gabungan bersama dengan kebijakan akses tingkat akun untuk memberikan akses ke sumber daya Anda AWS
- GAMESEC01-BP05 Gunakan penyedia identitas pusat

## GAMESEC01-BP01 Gunakan peran dan akses federasi, bukan pengguna root akun, untuk melakukan tindakan di lingkungan Anda AWS

Ketika Anda pertama kali membuat Akun AWS, Anda mulai dengan identitas yang dikenal sebagai pengguna root, yang diakses menggunakan alamat email dan kata sandi yang terkait dengan akun. Pengguna root memiliki akses lengkap ke AWS layanan dan sumber daya dalam akun itu. Dalam kebanyakan kasus, Anda harus menghindari menggunakan pengguna root untuk day-to-day tugas. Ketika akses tingkat root diperlukan, konfirmasikan bahwa itu mutlak diperlukan dan verifikasi bahwa logging dan pagar pembatas tambahan tersedia untuk melacak penggunaannya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Dalam AWS Organizations konfigurasi, setiap akun masih memiliki pengguna root sendiri, tetapi day-to-day akses harus dikelola melalui peran IAM dan pengguna IAM Identity Center. Buat akses berbasis peran yang disesuaikan dengan tahapan siklus hidup dan tim game Anda. Misalnya, tim operasi langsung mungkin memerlukan izin untuk mengelola acara dalam game, sementara pengembang memerlukan akses untuk mendorong pembaruan. Saat bekerja dengan layanan atau

mitra pihak ketiga, gunakan akses federasi untuk mengaktifkan kolaborasi aman tanpa mengekspos infrastruktur sensitif. Pendekatan ini memverifikasi bahwa setiap pengguna atau mitra hanya memiliki akses yang mereka butuhkan sambil menjaga keamanan infrastruktur dan data pemain game Anda.

Contoh pelanggan

AnyCompany Game menerapkan kontrol akses berbasis peran saat mengembangkan game baru mereka. Dengan menggunakan peran IAM khusus untuk tim pengembangan mereka yang beragam, mereka menghindari penggunaan kredensi bersama. Pengaturan ini memungkinkan tim pengembang untuk mengambil peran untuk sistem game inti, sedangkan peran tim konten hanya dapat mengakses layanan manajemen aset.

Langkah-langkah implementasi

- Jangan gunakan pengguna root setelah menyiapkan akun kecuali benar-benar diperlukan. Buat akun, amankan pengguna root, dan segera buat peran IAM administrasi yang diperlukan dan tetapkan peran itu ke pengguna federasi.
- Hanya gunakan pengguna root ketika Anda perlu melakukan sejumlah tugas terbatas yang hanya tersedia untuk pengguna root. Contoh tugas ini termasuk mengubah alamat email pengguna root Anda dan mengubah paket AWS dukungan Anda.

## GAMESEC01-BP02 Gunakan AWS Control Tower untuk mengatur lingkungan multi-akun dengan cepat AWS

Jika Anda mulai menggunakan hanya AWS dengan satu akun, Anda mungkin menemukan studio game Anda tumbuh darinya seiring kemajuan proses pengembangan game Anda. Misalnya, dengan satu Akun AWS, Anda mungkin mulai mencapai batas layanan, atau biaya Anda untuk berbagai proyek dan beban kerja mungkin menjadi lebih kompleks. Membuat akun berbeda untuk judul dan lingkungan game yang berbeda memungkinkan tim untuk bereksperimen dengan fitur baru, melewati batas layanan, dan mempertahankan postur dan kepatuhan keamanan. Dengan menerapkan strategi multi-akun di AWS, Anda bisa mendapatkan keuntungan dari mendistribusikan batas layanan di beberapa akun dan mendapatkan wawasan tentang biaya Anda. AWS

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Ini adalah kesalahpahaman umum bahwa menggunakan banyak secara otomatis Akun AWS akan lebih membingungkan dan memakan waktu. Sebaliknya, menggunakan AWS layanan yang

dirancang untuk memfasilitasi tata kelola beberapa akun dapat membantu studio game Anda menghabiskan lebih sedikit waktu mengelola akun Anda.

Anda dapat menggunakan layanan AWS Control Tower ini untuk menyediakan lingkungan multi-akun AWS dengan aman. Control Tower direkomendasikan jika Anda sedang membangun AWS lingkungan baru, memulai perjalanan Anda AWS, atau benar-benar baru AWS. Selama proses penyiapan singkat, Anda dapat berintegrasi dengan AWS layanan lain yang terlibat dengan pengelolaan akun dan akses pengguna, seperti, Service Catalog AWS Organizations, dan AWS IAM Identity Center.

Contoh pelanggan

AnyCompany Game awalnya dioperasikan dari satu Akun AWS, dan mereka mencapai beberapa hambatan ketika salah satu tim pengembangan game mereka mencapai batas EC2 layanan selama tes beta penting. Pada saat yang sama, tim pengembangan mereka untuk game yang berbeda berjuang dengan alokasi sumber daya untuk jalur pengujian otomatis mereka. Situasi mencapai titik puncak ketika AnyCompany Game tidak dapat secara akurat memisahkan biaya antar proyek, sehingga sulit untuk menganggarkan untuk pengembangan setiap game.

AnyCompany Game kemudian menerapkan strategi multi-akun menggunakan AWS Control Tower. Mereka membuat akun terpisah untuk setiap proyek game, dengan pengembangan, QA, dan lingkungan produksi yang berbeda. Pemisahan tingkat akun ini mengisolasi setiap data dan aset proyek, sehingga tim yang mengerjakan satu game tidak dapat mengakses atau memodifikasi sumber daya dari yang lain. Melalui AWS Organizations, mereka membentuk struktur penagihan terpusat yang dengan jelas menunjukkan biaya infrastruktur setiap game dan juga menciptakan kebijakan akses di seluruh organisasi.

Langkah-langkah implementasi

- Gunakan menara AWS Kontrol untuk mengatur lingkungan multi-akun otomatis.
- Atur akun berdasarkan lingkungan (seperti pengembangan, QA, dan produksi).
- Gunakan AWS IAM Identity Center dan Service Catalog untuk memusatkan izin pengguna dan merampingkan penyediaan sumber daya di seluruh akun.

## GAMESEC01-BP03 Gunakan kebijakan peran hak istimewa terkecil yang disesuaikan dengan fungsi pekerjaan tertentu

Mengkonfigurasi kebijakan IAM adalah bagian penting dari membangun fondasi keamanan yang kuat. Saat Anda menetapkan izin dengan kebijakan IAM, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Misalnya, tim QA memerlukan akses untuk mengubah hal-hal di lingkungan pengujian tetapi seharusnya tidak memiliki kemampuan untuk memodifikasi lingkungan produksi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Anda dapat memulai dengan izin luas, seperti kebijakan terkelola, saat Anda menjelajahi izin yang diperlukan untuk beban kerja atau kasus penggunaan. Saat kasus penggunaan Anda matang, Anda dapat bekerja untuk mengurangi izin yang Anda berikan untuk bekerja menuju hak istimewa yang paling sedikit.

Langkah-langkah implementasi

- Ikuti praktik izin hak istimewa paling sedikit untuk buat peran IAM untuk pengguna dan aplikasi.
- Gunakan kebijakan yang AWS dikelola untuk menyediakan akses luas dengan cepat sambil mengidentifikasi izin tertentu yang diperlukan tim atau aplikasi untuk melakukan tugas mereka.
- Studios juga dapat menggunakan pembuatan kebijakan penganalisis akses IAM untuk menghasilkan kebijakan IAM khusus berdasarkan CloudTrail peristiwa yang mengidentifikasi tindakan dan layanan yang digunakan oleh entri IAM.
- Secara teratur meninjau kebijakan IAM dan mengedit kebijakan yang terlalu permisif.

## GAMESEC01-BP04 Gunakan peran dan kebijakan akses gabungan bersama dengan kebijakan akses tingkat akun untuk memberikan akses ke sumber daya Anda AWS

AWS Pengguna baru sering menggunakan kebijakan IAM hanya ketika memberikan akses ke orang lain. Namun, jika Anda menggunakan AWS Organizations, pertimbangkan cara menggunakan kebijakan kontrol layanan bersama dengan kebijakan IAM untuk memberikan anggota tim studio Anda dan kontraktor tingkat akses yang diperlukan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Anda dapat membuat kebijakan IAM untuk mengizinkan atau menolak akses ke AWS layanan atau tindakan API yang bekerja dengannya AWS Identity and Access Management. Mereka hanya dapat diterapkan pada identitas IAM, seperti pengguna, grup, atau peran. Misalnya, kebijakan IAM dapat digunakan untuk menyediakan akses hanya-baca pengguna ke Amazon S3.

Kebijakan kontrol layanan (SCPs) adalah pagar pembatas untuk Anda. Akun AWS SCP tidak memberikan izin, mereka digunakan untuk membatasi tindakan pada AWS layanan untuk akun anggota individu. Misalnya, SCP dapat menolak Akun AWS mengakses Wilayah tertentu.

Ketika suatu tindakan diambil, kebijakan IAM yang relevan dievaluasi dalam kombinasi dengan. SCPs Menindaklanjuti contoh sebelumnya, adalah peran adalah upaya untuk menjalankan EC2 instance, IAM menunjukkan apakah mereka diizinkan (“Izinkan” untuk ec2:RunInstances) dan SCPs akan menentukan apakah pilihan Wilayah mereka valid (“us-east-1" diizinkan, tetapi “us-west-1" ditolak oleh SCP).

Melapisi kebijakan IAM dan SCPs dapat memverifikasi bahwa siapa pun yang mengakses AWS sumber daya Anda hanya akan diberikan izin yang sesuai yang mereka butuhkan. Ini sangat penting untuk dipertimbangkan jika Anda Akun AWS dan sumber daya menjangkau beberapa Wilayah, tetapi tidak semua orang di studio game Anda perlu mengakses semuanya.

Anda dapat menyesuaikan kebijakan IAM untuk memberikan izin khusus kepada tim tertentu untuk memperbarui hal-hal seperti konfigurasi game, mengelola data pemain, mengonfigurasi acara promosi, dan memoderasi konten buatan pengguna. Sementara itu, Anda dapat menggunakan SCPs untuk menegakkan kontrol seluruh organisasi yang penting untuk operasi game. Ini mungkin termasuk membatasi penyebaran hanya ke Wilayah yang disetujui tempat game beroperasi, membantu mencegah akses tidak sah ke penyimpanan data pemain yang sensitif, menegakkan persyaratan kepatuhan, dan mengendalikan biaya dengan membatasi penggunaan layanan di seluruh akun pengembangan.

Langkah-langkah implementasi

- Gunakan kebijakan IAM untuk mengelola izin bagi pengguna, grup, atau peran individu.
- Gunakan kebijakan kontrol layanan (SCPs) AWS Organizations untuk menerapkan izin tingkat akun.
- Gabungkan kebijakan IAM dan SCPs hanya memberikan akses yang diperlukan untuk pengguna dan akun tertentu.

Sumber daya

- Kebijakan dan izin di AWS IAM
- Kebijakan kontrol layanan
- Kebijakan terkelola AWS untuk fungsi pekerjaan

## GAMESEC01-BP05 Gunakan penyedia identitas pusat

Penyedia identitas pusat bertindak sebagai sumber tunggal untuk menyimpan dan mengelola kredensi pengguna, identitas, izin, dan otentikasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Gunakan penyedia identitas pusat untuk merampingkan proses otentikasi pengguna Anda, menerapkan kebijakan keamanan yang konsisten, dan menyederhanakan manajemen pengguna di seluruh aplikasi Anda. Akun AWS Memiliki pendekatan terpusat menghilangkan kebutuhan untuk mengelola identitas dan kredensional pengguna secara terpisah, yang mengurangi risiko inkonsistensi, redundansi, dan kerentanan keamanan lainnya. Menggabungkan identitas pengguna dan otentikasi ke satu tempat juga memungkinkan visibilitas, kontrol, dan auditabilitas yang lebih baik untuk seluruh lingkungan Anda. AWS

Contoh pelanggan

AnyCompany Game menghadapi tantangan signifikan dengan mengelola akses pengembang di seluruh AWS infrastruktur mereka yang berkembang pesat. Tim pengembangan mereka tumbuh dari 50 menjadi 200 orang di tiga judul utama. Awalnya, setiap tim proyek mengelola kredensi AWS akses mereka sendiri, menghasilkan praktik keamanan yang tidak konsisten, keterlambatan orientasi untuk pengembang baru, dan insiden keamanan sesekali.

Studio menerapkan AWS IAM Identity Center sebagai penyedia identitas pusat mereka, mengkonsolidasikan manajemen pengguna ke dalam satu sistem. Mereka menghubungkannya dengan direktori perusahaan yang ada, memungkinkan pengembang untuk menggunakan kredensi perusahaan yang sama untuk akses. AWS Sekarang pengembang menggunakan login perusahaan tunggal mereka yang ada untuk mendapatkan AWS akses yang mereka butuhkan untuk menyelesaikan pekerjaan mereka

Langkah-langkah implementasi

- Pertimbangkan untuk menggunakan AWS IAM Identity Center sebagai penyedia identitas pusat Anda. Ini memberikan manajemen akses yang konsisten di seluruh Anda Akun AWS, memberi karyawan Anda otentikasi masuk tunggal, dan menyederhanakan audit akses pengguna ke aplikasi Anda. AWS IAM Identity Center juga terhubung dengan identitas perusahaan yang ada dari penyedia identitas yang didukung.

## Keamanan yang sedang berlangsung

GAMESEC02: Bagaimana Anda mencapai, memelihara, dan memantau praktik terbaik keamanan yang sedang berlangsung?

Mengikuti praktik keamanan terbaik sangat penting untuk bisnis di seluruh industri, tetapi terutama di industri game. Industri game bergantung pada menumbuhkan dan mempertahankan kepercayaan pemain dan menciptakan reputasi yang kuat, dan bahkan masalah keamanan kecil dapat dengan cepat merusak kepercayaan itu.

Selain itu, sifat global industri game mengharuskan kepatuhan terhadap berbagai peraturan dan standar industri yang mengatur perlindungan data, privasi konsumen, dan keamanan di seluruh Wilayah tempat game ditawarkan. Gameplay yang adil dan aman adalah aspek penting lain yang menggarisbawahi pentingnya langkah-langkah keamanan yang kuat. Kecurangan, peretasan, dan bentuk eksploitasi game lainnya dapat mengganggu pengalaman bermain game untuk pemain yang sah, yang membuat kontrol keamanan yang kuat penting untuk menjaga integritas gameplay dan menumbuhkan lapangan bermain yang setara bagi peserta.

Praktik terbaik

- GAMESEC02-BP01 Gunakan template siap pakai untuk praktik keamanan standar
- GAMESEC02-BP02 Gunakan teknik remediasi otomatis saat peristiwa keamanan muncul

# GAMESEC02-BP01 Gunakan template siap pakai untuk praktik keamanan standar

Ready-to-deploytemplate menyediakan cara proaktif dan gesit untuk menilai postur keamanan Anda di cloud. Template yang telah dikonfigurasi sebelumnya mengevaluasi keamanan cloud Anda dan segera menerapkan perubahan yang diperlukan. Template mencakup berbagai praktik terbaik di berbagai teknologi dan kerangka kerja keamanan yang diterima secara luas. Menggunakan template dapat membantu studio game untuk mempertahankan konfigurasi infrastruktur yang konsisten, terutama karena mereka dapat menskalakan dan menambahkan tambahan Akun AWS untuk mendukung beban kerja baru.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Dengan menggunakan AWS layanan dan menerapkan ready-to-deploy template, pengembang game dapat secara proaktif menilai dan memperkuat postur keamanan cloud mereka, menjaga kekayaan intelektual mereka, melindungi data pemain, dan membina lanskap game yang aman melalui penilaian keamanan reguler dan pemantauan berkelanjutan untuk segera mengidentifikasi dan mengatasi potensi kerentanan.

Contoh pelanggan

AnyCompany Game menghadapi tantangan yang signifikan ketika bersiap untuk meluncurkan gelar berikutnya di industri Eropa. Mereka menyadari bahwa praktik penanganan data yang ada tidak memenuhi persyaratan GDPR. Mereka beralih ke AWS Security Hub CSPM dan AWS Config dan ready-to-deploy templatnya untuk solusi. Tim menerapkan paket kesesuaian khusus GDPR di AWS Config, yang secara otomatis menilai infrastruktur mereka yang ada terhadap standar GDPR. Pemindaian awal ini mengungkapkan beberapa celah penting, seperti kebijakan retensi data yang tidak tepat dan kontrol akses yang tidak memadai di mana data pemain disimpan. Menggunakan aturan template yang telah ditentukan, AnyCompany Game dengan cepat menerapkan perubahan yang diperlukan. Selain itu, pemeriksaan kepatuhan otomatis yang sedang berlangsung yang disediakan oleh template memungkinkan tim kecil untuk mempertahankan kepatuhan GDPR dengan mudah, bahkan ketika mereka terus memperbarui dan memperluas permainan.

Langkah-langkah implementasi

- Gunakan templat untuk praktik keamanan standar, seperti aturan terkelola dan paket kesesuaian AWS Config dan standar di. AWS Security Hub CSPM

- Tinjau detail standar CSPM Security Hub untuk menentukan mana yang paling sesuai dengan kebutuhan keamanan studio game Anda.

## GAMESEC02-BP02 Gunakan teknik remediasi otomatis saat peristiwa keamanan muncul

Dengan menggunakan teknik remediasi otomatis, pengembang game dapat secara proaktif melindungi dan memelihara infrastruktur game mereka dan meminimalkan dampak potensial yang mungkin ditimbulkan oleh insiden keamanan. Jika masalah keamanan terdeteksi, gunakan runbook untuk memandu respons Anda terhadap situasi tersebut. Otomatiskan tanggapan ini jika memungkinkan untuk memulihkan masalah lebih cepat dan mengurangi dampaknya. Ini meningkatkan pengalaman pemain dengan mengurangi kemungkinan downtime dan gangguan pada permainan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Bersiap untuk menanggapi masalah keamanan tidak hanya melindungi pengalaman para pemain tetapi juga untuk memenuhi berbagai standar kepatuhan dan peraturan. Selain itu, menggunakan respons keamanan otomatis menskalakan operasi keamanan Anda saat beban kerja Anda bertambah. AWS menyediakan layanan untuk membantu mengidentifikasi dan mengotomatiskan respons terhadap insiden ini.

Contoh pelanggan

AnyCompany Game menghadapi insiden keamanan kritis ketika ember S3 yang berisi model karakter dan tekstur yang belum dirilis untuk game mendatang mereka secara tidak sengaja dipublikasikan selama pembaruan pipa aset rutin. Sistem keamanan otomatis mendeteksi perubahan izin bucket dalam beberapa menit setelah modifikasi. Sistem segera mengeksekusi runbook remediasinya: mengembalikan bucket ke status pribadi, mencatat upaya akses selama jendela eksposur, memberi tahu tim keamanan, dan membuat CloudTrail log terperinci tentang perubahan izin.

Langkah-langkah implementasi

- Gunakan Respons Keamanan Otomatis pada AWS solusi untuk mengimplementasikan runbook otomatisasi yang menentukan tindakan yang akan diambil secara otomatis sebagai respons terhadap peristiwa keamanan di AWS Security Hub CSPM.

Sumber daya

- AWS untuk Blog Game - Mengelola Studio Game Anda di AWS: Bagian 1
- AWS untuk Blog Game - Mengelola Studio Game Anda di AWS bagian 2
- Daftarkan unit organisasi yang ada dengan AWS Control Tower
- Akun AWS pengguna root
- Tugas yang membutuhkan kredensil pengguna root
- Respon Keamanan Otomatis pada AWS

# Manajemen identitas dan akses

GAMESEC03: Bagaimana Anda mengelola identitas pemain dan manajemen akses?

Saat mengembangkan game, Anda harus menentukan bagaimana Anda akan memberi pemain akses ke permainan Anda dan layanan terkait. Bagian berikut mengeksplorasi pertimbangan desain, termasuk otentikasi pemain, otorisasi, dan otentikasi multi-faktor.

Praktik terbaik

- GAMESEC03-BP01 Tentukan pendekatan Anda untuk mengidentifikasi dan mengontrol akses pemain ke lingkungan dan sumber daya game Anda
- GAMESEC03-BP02 Otentikasi permintaan yang dikirim ke layanan backend game Anda
- GAMESEC03-BP03 Gunakan layanan backend game Anda untuk memvalidasi permintaan pemain untuk bergabung dengan game multipemain
- GAMESEC03-BP04 Menegakkan kebijakan keamanan yang ketat untuk akun pengguna pemain dengan membutuhkan kata sandi yang kuat
- GAMESEC03-BP05 Menyediakan opsi bagi pemain untuk mengatur otentikasi multi-faktor (MFA) di akun mereka

# GAMESEC03-BP01 Tentukan pendekatan Anda untuk mengidentifikasi dan mengontrol akses pemain ke lingkungan dan sumber daya game Anda

Keputusan ini dipengaruhi oleh strategi akuisisi dan monetisasi pemain Anda, pengalaman pemain, dan faktor-faktor lain seperti kemampuan yang ada yang mungkin disediakan oleh mitra penerbitan game Anda. Misalnya, permainan mungkin memerlukan pembelian dan mengharuskan pemain untuk membuat profil pengguna untuk mengaitkan metode pembayaran uang nyata dengan akun mereka.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Atau, permainan mungkin ingin mengurangi penghalang masuk untuk pengalaman pemain pertama kali dengan menghapus kebutuhan untuk membuat akun pengguna sebelum bermain game, sehingga meningkatkan kemungkinan pemain akan mencoba permainan untuk pertama kalinya. Biasanya, game akan menerapkan satu atau lebih kombinasi identitas pemain dan pendekatan manajemen akses untuk permainan mereka.

Akses tidak diautentikasi atau anonim

Tingkat akses ini berguna dalam situasi di mana permainan tidak mengharuskan pemain untuk membuat akun pengguna baru atau menautkan dengan identitas mereka di jejaring sosial dan sistem game. Ini adalah cara paling sederhana dan tercepat bagi pemain untuk mulai bermain game dan sangat berguna dalam game seluler di mana pengembang game mungkin ingin mengurangi hambatan masuk untuk pengalaman awal.

Dalam skenario akses ini, jika Anda ingin mengidentifikasi penggunaan dari instalasi game, Anda dapat memprogram klien game untuk menghasilkan dan menyimpan pengenal unik ke perangkat pemain. Pengenal unik ini digunakan untuk mengidentifikasi pemain di seluruh sesi permainan di perangkat mereka dan memungkinkan pelaporan analitik tentang penggunaan dari waktu ke waktu. Kemudian, jika pemain memilih untuk membuat akun, Anda dapat mengaitkan akun pengguna baru mereka dengan pengenal unik yang dibuat sebelumnya. Ini akan menghubungkan identitas pemain baru mereka dengan penggunaan historis mereka, yang mungkin termasuk statistik dan pencapaian game.

Jika seorang pemain pada akhirnya tidak membuat dan menautkan akun, perangkat yang digunakan pemain untuk berinteraksi dengan permainan dapat diidentifikasi secara unik, tetapi informasi yang dapat dipulihkan tentang pemain tidak dikumpulkan dan disimpan. Jadi, jika pemain merusak atau

kehilangan perangkat mereka, data yang disimpan sebelumnya yang terkait dengan perangkat juga hilang dan mungkin tidak dapat dipulihkan.

Otentikasi dengan nama pengguna dan kata sandi

Sebuah permainan memungkinkan pemain untuk membuat akun pengguna mereka sendiri dengan nama pengguna dan kata sandi yang disimpan dalam backend game. Ini mungkin terjadi ketika pengembang game berkolaborasi dengan penerbit game yang sudah memiliki sistem akun pemain yang sudah ada yang dapat diintegrasikan oleh pengembang. Atau, pengembang yang menerbitkan game mereka sendiri mungkin ingin menyederhanakan pengalaman pemain dengan memungkinkan pemain membuat akun pengguna tunggal untuk akses di seluruh game yang mereka terbitkan.

Otentikasi dan penautan akun dengan jejaring sosial pihak ketiga dan sistem game

Adalah umum untuk game online dan game dengan fitur sosial untuk menyediakan federasi penyedia identitas pihak ketiga untuk menyederhanakan pengalaman pemain. Alih-alih meminta pemain untuk membuat kombinasi nama pengguna dan kata sandi untuk mengautentikasi, Anda dapat menggunakan federasi identitas untuk memungkinkan pemain mengautentikasi menggunakan akun pihak ketiga mereka dengan jejaring sosial dan sistem game. Proses login ini menyederhanakan pengalaman masuk dan pendaftaran untuk pemain. Ini juga menyediakan alternatif yang nyaman untuk pembuatan akun wajib dan metode tanpa gesekan bagi pemain untuk mengakses game.

Untuk pengembang game, proses login federasi dapat menawarkan alur kerja verifikasi pemain yang efisien. Ini juga dapat memberikan cara yang lebih andal untuk mengelola data pemain yang digunakan untuk personalisasi. Ini karena Anda tidak perlu meminta pemain untuk memberi Anda data tertentu yang kemungkinan telah mereka berikan kepada penyedia identitas pihak ketiga. Selain itu, sistem ini menyediakan integrasi dengan fitur sosial tambahan seperti kemampuan untuk menghubungkan pemain dengan teman-teman mereka.

Langkah-langkah implementasi

- Gunakan akses tidak terautentikasi atau anonim untuk mengurangi hambatan bagi pemain pertama kali dengan membuat pengenal perangkat unik untuk melacak penggunaan dan memungkinkan penautan akun nanti.
- Menerapkan otentikasi nama pengguna dan kata sandi untuk akun pengguna khusus, menggunakan sistem akun pemain yang ada atau menciptakan pengalaman terpadu di seluruh game.
- Integrasikan penyedia identitas pihak ketiga untuk otentikasi federasi, menyederhanakan proses login dan memungkinkan akses ke fitur sosial dan data personalisasi.

# GAMESEC03-BP02 Otentikasi permintaan yang dikirim ke layanan backend game Anda

Mengautentikasi permintaan yang dikirim ke layanan backend game Anda dapat memblokir permintaan yang tidak diinginkan agar tidak berhasil.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Anda harus menyediakan layanan otentikasi bagi pemain untuk masuk, yang harus mengembalikan token berumur pendek yang aman, seperti JSON Web Token (JWT), ke klien game ketika pemain berhasil mengautentikasi.

Token ini dapat mencakup pernyataan klaim yang berisi atribut pemain dan metadata relevan lainnya. Metadata yang relevan ini dapat digunakan dalam permintaan berikutnya yang dikirim dari klien game ke backend game Anda untuk mengotentikasi permintaan dan mengotorisasi mereka dalam konteks pemain yang diautentikasi.

Anda memiliki opsi untuk merancang dan membangun sistem otentikasi pemutar Anda sendiri, yang memerlukan peningkatan dan pemeliharaan berkelanjutan, atau Anda dapat menggunakan fitur pendaftaran, masuk, dan kontrol akses pengguna yang dapat diskalakan dan aman yang disediakan oleh Amazon Cognito.

Kumpulan pengguna Amazon Cognito menyertakan direktori pengguna untuk otentikasi dan otorisasi. Kumpulan pengguna menyediakan APIs bahwa Anda dapat mengintegrasikan ke dalam game Anda untuk alur kerja pendaftaran, masuk, dan mengatur ulang kata sandi, yang dapat diintegrasikan dengan penyedia identitas pihak ketiga. Application Load Balancers dan Amazon API Gateway keduanya menyediakan integrasi dengan Cognito untuk mengintegrasikan otentikasi pengguna untuk permintaan yang dikirim ke backend game kustom Anda yang dihosting dengan layanan ini.

Jika game Anda mendukung akses anonim dan Anda tidak dapat mengautentikasi pemain, Anda dapat menggunakan pendekatan otentikasi klien untuk memberikan pengalaman yang lebih aman saat mengintegrasikan dengan backend game Anda. Jika klien game Anda menggunakan AWS layanan secara langsung, permintaan untuk layanan ini harus ditandatangani menggunakan kredensi. Untuk memberikan kredensi kepada klien game Anda bagi pengguna yang tidak diautentikasi, Anda dapat menggunakan AWS SDK untuk mengambil kredenal berumur pendek dari kumpulan identitas Amazon Cognito yang dapat digunakan untuk menandatangani permintaan Anda ke layanan. AWS Kredensi ini dapat disegarkan dari klien game Anda.

Selain terintegrasi langsung dengan AWS SDK dari klien game, Anda juga dapat membuat backend game Anda sendiri, menggunakan layanan seperti Amazon API Gateway, yang mendukung otorisasi khusus. Dengan merancang layanan backend game Anda sendiri, Anda dapat memperoleh kontrol otoritatif atas permintaan dengan logika sisi server khusus.

Untuk informasi selengkapnya tentang membangun layanan backend untuk game yang dihosting menggunakan Amazon GameLift, lihat Merancang layanan klien game Anda.

Contoh pelanggan

AnyCompany Game meningkatkan keamanan judul berikutnya dengan mengadopsi pendekatan otentikasi dan otorisasi terkelola. Alih-alih mempertahankan sistem nama pengguna dan kata sandi khusus, mereka menggunakan kumpulan pengguna Amazon Cognito untuk menangani pendaftaran dan masuk pemain, dan kumpulan identitas untuk mendukung akses anonim bagi pemain yang mencoba mode pelatihan sebelum membuat akun. Mereka juga menerapkan logika otorisasi khusus dalam game untuk mengenali peran administrator yang ditentukan dalam Cognito, memberikan pengguna tersebut akses ke fitur manajemen dalam game khusus.

Langkah-langkah implementasi

- Gunakan kumpulan pengguna Amazon Cognito untuk mengelola otentikasi dengan token aman seperti JWTs, mengaktifkan fitur seperti pendaftaran, masuk, dan pengaturan ulang kata sandi.
- Ambil kredensi berumur pendek dari kumpulan identitas Amazon Cognito agar pengguna anonim dapat berinteraksi secara aman dengan layanan. AWS
- Menerapkan backend game kustom menggunakan Amazon API Gateway untuk logika otentikasi sisi server kustom.

## GAMESEC03-BP03 Gunakan layanan backend game Anda untuk memvalidasi permintaan pemain untuk bergabung dengan game multipemain

Biasanya, dalam game multipemain, pemain akan bergabung dengan sesi permainan dengan memilih opsi langsung dari daftar sesi yang tersedia, atau mereka akan mengirimkan permintaan untuk menemukan kecocokan. Pendekatan terakhir menempatkan tanggung jawab pada pengembang game untuk menemukan sesi permainan yang memenuhi syarat dan memberikan informasi koneksi (biasanya alamat IP dan nomor port) kembali ke klien game pemain. Implementasinya dapat bervariasi tergantung pada genre game yang Anda kembangkan, tetapi

terlepas dari itu, ini adalah praktik terbaik keamanan untuk melakukan validasi sisi server dari permintaan pemain untuk bergabung dengan game.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Misalnya, dalam permainan multipemain berbasis sesi, permintaan dari pemain untuk bergabung dengan sesi permainan harus divalidasi oleh perangkat lunak server game Anda dengan layanan perjodohan backend game Anda sebelum mengotorisasi koneksi mereka ke server. Ketika seorang pemain meminta untuk bergabung dengan sesi permainan, server game harus memeriksa permintaan untuk pengenal unik, seperti ID sesi pemain dan tiket yang dibuat server yang sebelumnya diberikan kepada klien game oleh layanan perjodohan backend game Anda.

Setelah memulai koneksi ke server game, perangkat lunak sisi server Anda dapat menggunakan informasi ini untuk memverifikasi dengan layanan perjodohan bahwa permintaan koneksi pemain valid dan memverifikasi bahwa pemain tidak bergabung dengan tempat yang sebelumnya dicadangkan dalam sesi permainan untuk pemain lain.

Untuk game yang di-host di Amazon GameLift, lihat client/server Interaksi game dengan GameLift Server Amazon untuk contoh bagaimana jenis validasi sisi server ini dapat diterapkan.

### Contoh pelanggan

Selama peluncuran beta AnyCompany awal Game, mereka menemukan bahwa pemain melewati sistem perjodohan mereka dengan langsung terhubung ke server game, yang mengarah ke masalah integritas kompetitif yang serius. Ketika pemain berperingkat tinggi menemukan bahwa mereka dapat berbagi alamat IP server dengan teman-teman, mereka mulai menghindari sistem perjodohan berbasis keterampilan, menghasilkan pemain berpengalaman bergabung dengan pertandingan pemula dan menciptakan pengalaman yang membuat frustrasi bagi pemain baru. AnyCompany Game merespons dengan menerapkan sistem validasi sisi server yang menghasilkan tiket sesi unik untuk setiap permintaan perjodohan. Sistem ini membutuhkan tiket permintaan pemain IDs dan perjodohan dan upaya koneksi terverifikasi terhadap layanan perjodohan backend mereka.

### Langkah-langkah implementasi

- Validasi permintaan gabungan pemain di sisi server menggunakan pengidentifikasi unik seperti sesi pemain dan tiket yang dibuat server. IDs
- Konfirmasikan validitas permintaan koneksi dengan layanan perjodohan untuk memblokir akses yang tidak sah.

- Verifikasi bahwa tempat yang dipesan dalam sesi permainan tidak diakses oleh pemain yang tidak sah selama proses validasi.

## GAMESEC03-BP04 Menegakkan kebijakan keamanan yang ketat untuk akun pengguna pemain dengan membutuhkan kata sandi yang kuat

Jika permainan memberi pemain kemampuan untuk membuat akun pengguna dengan kata sandi, Anda harus meminta kata sandi pemain untuk mematuhi kebijakan yang kuat. Misalnya, kumpulan pengguna Amazon Cognito memberi Anda kemampuan untuk menentukan persyaratan kata sandi untuk akun pengguna. Membuat kebijakan kata sandi yang kuat dapat melindungi akun pemain Anda agar tidak diambil alih melalui rekayasa sosial dan serangan brute force.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Contoh pelanggan

AnyCompany Game menghadapi krisis ketika judul populer mereka mengalami gelombang pembajakan akun karena kebijakan kata sandi yang lemah. Pemain yang menggunakan kata sandi sederhana seperti “password123" menjadi korban serangan brute force otomatis, mengakibatkan item hilang dan mata uang dalam game dikompromikan. Untuk mengatasi hal ini, AnyCompany Game mengubah sistem login mereka dan mengamanatkan bahwa kata sandi tidak digunakan sebelumnya, termasuk setidaknya satu huruf besar, satu angka, satu karakter khusus, dan panjang minimum 15 karakter.

Langkah-langkah implementasi

- Memerlukan kebijakan kata sandi yang kuat untuk akun pemain untuk meningkatkan keamanan.
- Gunakan kumpulan pengguna Amazon Cognito untuk menentukan dan menerapkan persyaratan kata sandi.

## GAMESEC03-BP05 Menyediakan opsi bagi pemain untuk mengatur otentikasi multi-faktor (MFA) di akun mereka

Akun pemain dapat menjadi aset bagi pelaku jahat, terutama dalam game yang mendukung mata uang dan pembelian dalam game. Karena meluasnya peretasan akun pemain dan serangan

rekayasa sosial, berikan pemain opsi untuk meningkatkan keamanan akun mereka dengan mengonfigurasi otentikasi multi-faktor (MFA).

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Ketika seorang pemain mencoba mengakses akun mereka dengan menggunakan MFA, kode sementara dikirim ke alamat email, nomor telepon, atau aplikasi seluler otentikasi multi-faktor yang dibuat khusus. Agar berhasil mengautentikasi, pemain kemudian harus memasukkan kode ke dalam sistem login dalam jangka waktu terbatas.

MFA juga dapat digunakan untuk membantu melindungi akun yang mencoba mengautentikasi dari lokasi geografis baru, akun yang telah ditandai oleh dukungan pemain untuk potensi aktivitas berbahaya, dan bahkan untuk akun yang belum masuk ke game untuk waktu yang lama.

Misalnya, kumpulan pengguna Amazon Cognito dapat mengonfigurasi otentikasi multi-faktor pada direktori pengguna.

Langkah-langkah implementasi

- Aktifkan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun pemain.
- Gunakan kode sementara yang dikirim melalui email, telepon, atau aplikasi MFA untuk memverifikasi akses akun.
- Terapkan MFA untuk geo-lokasi baru, akun yang ditandai, atau akun dengan ketidakaktifan yang diperpanjang.

## Kontrol akses

GAMESEC04: Bagaimana Anda memblokir akses tidak sah ke konten game?

Game modern mencakup sejumlah besar konten, seperti konten yang dapat diunduh (DLC), yang merupakan aspek penting dari keterlibatan pemain dan monetisasi game. Pemain mengharapkan aliran karakter, level, dan tantangan baru yang berkelanjutan, yang mengharuskan pengembang game untuk mengikuti permintaan konstan akan konten segar untuk mempertahankan pemain.

Variasi dan ukuran konten dapat sangat bervariasi tergantung pada jenis permainan dan perangkat tempat permainan dimainkan. Terlepas dari sistem game, lindungi konten game dari akses yang tidak sah.

Praktik terbaik

- GAMESEC04-BP01 Membatasi akses konten yang dapat diunduh ke klien dan pengguna yang berwenang
- GAMESEC04-BP02 Batasi akses asal ke jaringan pengiriman konten resmi () CDNs
- GAMESEC04-BP03 Menerapkan pembatasan geografis untuk membatasi akses yang tidak sah
- GAMESEC04-BP04 Membatasi akses ke konten dengan solusi manajemen hak digital (DRM)

## GAMESEC04-BP01 Membatasi akses konten yang dapat diunduh ke klien dan pengguna yang berwenang

Batasi akses ke konten game oleh aplikasi dan klien resmi. Pertimbangkan untuk menggunakan Amazon S3 sebagai sumber yang hemat biaya dan dapat diskalakan untuk menyimpan konten game yang dapat diunduh dan CloudFront Amazon untuk menyediakan pengiriman konten berkinerja global kepada pemain. Kedua layanan menyediakan mekanisme bawaan untuk membatasi akses ke data yang disimpan, seperti membatasi akses ke pengguna yang diautentikasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Memberikan akses ke konten yang disimpan di Amazon S3

Ketika Anda perlu memberikan akses ke konten yang disimpan di S3, ada beberapa faktor yang perlu dipertimbangkan. Secara default, hanya Akun AWS yang membuat bucket S3 yang dapat mengakses objek yang tersimpan di dalamnya. Untuk memberikan akses ke aplikasi internal Anda dan mengelola konten yang disimpan di bucket Amazon S3, gunakan AWS Identity and Access Management (IAM) untuk membuat kebijakan yang menyediakan akses yang sesuai.

Peran IAM dapat dikaitkan dengan pengguna, sistem, atau aplikasi federasi yang dihosting di layanan, seperti Amazon EC2,, dan aplikasi berbasis kontainer yang dihosting di Amazon EKS dan Amazon ECS. AWS Lambda Misalnya, Anda dapat menggunakan AWS SDK atau AWS CLI untuk memublikasikan dan mengelola aset konten game di bucket S3. Untuk mendukung kasus penggunaan ini, Anda dapat membuat peran IAM dengan akses yang sesuai untuk membaca

dan menulis konten game ke bucket S3 Anda dan mengaitkannya dengan EC2 instance yang menghosting perangkat lunak dan skrip Anda.

Kebijakan berbasis sumber daya dapat ditentukan untuk bucket Anda dan untuk objek tertentu. Kebijakan bucket S3 dikaitkan dengan bucket S3 dan dapat digunakan untuk membatasi akses ke bucket dan objek di dalamnya, serta memberikan akses ke sumber daya Amazon S3 Anda dari akun lain. Misalnya, dalam skenario di mana beberapa tim atau studio pengembangan game terpisah mengerjakan konten game yang sama dan memerlukan akses yang sama ke konten yang dihosting secara terpusat di Amazon S3, Anda dapat menggunakan kebijakan bucket S3 untuk menentukan izin akses lintas akun ke sumber daya S3. Pertimbangkan untuk menggunakan titik akses S3, yang dapat menyederhanakan pengelolaan akses data ke data bersama dengan membuat titik akses dengan nama dan izin khusus untuk setiap aplikasi atau set aplikasi. Dokumentasi Amazon S3 berisi praktik terbaik tambahan untuk kontrol akses di Amazon S3.

**Granting short-term access to your content**

Ketika akses hanya diperlukan untuk waktu terbatas tertentu, buat sementara URLs yang memberikan akses jangka pendek ke konten Anda. Amazon S3 menyediakan dukungan untuk menghasilkan presigned URLs, yang memungkinkan pemilik objek memberikan akses terbatas waktu ke objek di Amazon S3 tanpa memperbarui kebijakan bucket Anda. Dengan demikian, pengguna akhir atau aplikasi yang diberikan akses tidak diharuskan memiliki akun atau izin IAM dan sebagai gantinya menggunakan URL yang telah ditetapkan sebelumnya untuk mengakses konten.

Ini adalah praktik terbaik yang biasa digunakan dalam berbagai kasus penggunaan game, seperti memberikan akses kepada pemain yang berwenang ke konten yang dapat diunduh yang berhak mereka dapatkan dan menyediakan akses sementara ke konten game dengan waktu terbatas. Presigned juga URLs dapat digunakan untuk memberikan izin sementara untuk mengunggah konten ke bucket S3. Misalnya, Anda dapat mempertimbangkan untuk menggunakan URL yang telah ditetapkan sebelumnya untuk memberi pemain akses untuk mengunggah log klien untuk membantu tim dukungan Anda memecahkan masalah kasus dukungan pemain.

Menggunakan jaringan pengiriman konten untuk menyediakan akses ke konten Anda

Meskipun aplikasi, pengembang game, artis, dan personel lainnya mungkin memerlukan akses langsung ke konten di bucket S3 untuk tujuan pengembangan dan manajemen, gunakan jaringan pengiriman konten untuk menyediakan akses ke konten yang tersedia untuk umum bagi pemain atau pengguna lain melalui internet. Pendekatan ini meningkatkan kinerja unduhan dan mengurangi biaya dengan menyimpan konten yang sering diakses. Amazon CloudFront dapat mendistribusikan konten

Anda secara global dengan melakukan cache dan mengirimkannya lebih dekat ke pemain Anda sekaligus mengurangi beban pada asal unduhan game Anda, seperti Amazon S3.

Daripada menyajikan konten publik Anda langsung dari bucket S3, disarankan untuk menjaga konten ini tetap pribadi dan menyajikannya secara publik dengan menggunakan. CloudFront CloudFront dapat dikonfigurasi untuk mengharuskan pemain mengakses konten pribadi Anda (seperti unduhan game baru hanya untuk pemain berbayar) dengan menggunakan cookie yang ditandatangani URLs atau ditandatangani. Anda kemudian dapat mengembangkan aplikasi Anda baik untuk membuat dan mendistribusikan yang ditandatangani URLs ke pengguna yang diautentikasi, atau untuk mengirim header set-cookie yang menetapkan cookie yang ditandatangani untuk pengguna yang diautentikasi. Saat Anda membuat cookie yang ditandatangani URLs atau ditandatangani untuk mengontrol akses ke file Anda, Anda dapat menentukan tanggal dan waktu berakhir, setelah itu URL dan cookie tidak lagi valid.

Secara opsional, Anda juga dapat menentukan alamat IP atau rentang alamat komputer yang dapat digunakan untuk mengakses konten Anda, yang berguna jika Anda ingin membatasi akses ke mitra studio pengembangan game tertentu atau jaringan kontraktor. Gunakan cookie yang ditandatangani saat Anda ingin memberikan akses ke beberapa file yang dibatasi, atau jika Anda tidak ingin mengubah file Anda saat ini URLs. Gunakan tanda tangan URLs saat Anda ingin membatasi akses ke file individual atau jika pengguna Anda menggunakan klien yang tidak mendukung cookie. Ditandatangani URLs lebih diutamakan daripada cookie yang ditandatangani.

Langkah-langkah implementasi

- Gunakan peran IAM dan kebijakan bucket untuk memberikan akses yang sesuai ke bucket S3 untuk aplikasi internal, tim, atau skenario lintas akun.
- Hasilkan presigned URLs untuk memberikan akses jangka pendek ke objek S3, cocok untuk konten yang dapat diunduh atau unggahan sementara seperti log klien.
- Gunakan Amazon CloudFront dengan tanda tangan URLs atau cookie untuk menyajikan konten pribadi dengan lebih aman kepada pengguna yang diautentikasi

## GAMESEC04-BP02 Batasi akses asal ke jaringan pengiriman konten resmi () CDNs

Blokir pengguna agar tidak menghindari jaringan pengiriman konten Anda untuk langsung mengakses konten dari asal Anda, seperti bucket Amazon S3 Anda. Penting untuk membatasi akses ke asal Anda hanya untuk yang berwenang CDNs, yang mengurangi biaya transfer data dari

penyajian konten yang tidak perlu di luar asal. Ini juga meningkatkan postur keamanan Anda dengan mengalirkan akses publik ke konten asal Anda melalui titik masuk yang sama, di mana Anda dapat menerapkan kontrol keamanan tepi seperti pemfilteran AWS WAF lapisan 7, injeksi dan inspeksi parameter permintaan HTTP terkait keamanan, dan perlindungan penolakan layanan terdistribusi (S). DDo

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Untuk menerapkan kontrol ini untuk asal Amazon S3, Anda dapat menggunakan identitas akses CloudFront asal Amazon (OAI), yang memverifikasi bahwa permintaan ke objek S3 Anda berasal dari distribusi Anda. CloudFront Kaitkan AWS WAF dengan CloudFront distribusi Anda untuk menyediakan penyaringan layer-7. Namun, jika Anda menyajikan konten dari tambahan CDNs, Anda dapat mengonfigurasi CDN untuk memasukkan satu atau lebih header HTTP kustom ke permintaan asal yang dapat diperiksa oleh AWS WAF untuk memverifikasi bahwa lalu lintas masuk berasal dari penyedia CDN resmi Anda.

Pendekatan ini juga berguna untuk membantu mencegah pengguna menghindari penyedia CDN Anda ketika asal Anda di-host di belakang Application Load Balancer (ALB). ALBs dapat dikaitkan dengan AWS WAF untuk perlindungan lapisan-7. Anda dapat mengonfigurasi AWS WAF untuk menyisipkan header HTTP kustom yang akan diperiksa oleh ALB Anda untuk memproses dan memeriksa lalu lintas masuk ke penyeimbang beban oleh. AWS WAF

Contoh pelanggan

AnyCompany Game menerapkan pembatasan akses asal untuk membantu melindungi aset game mereka, konten yang dapat diunduh, dan file patch dari akses langsung yang tidak sah yang dapat memungkinkan pemain untuk melewati pemeriksaan keamanan atau mendapatkan konten premium tanpa otentikasi yang tepat. Pendekatan ini memungkinkan mereka untuk memantau pola akses konten melalui titik terpusat, sehingga mudah bagi mereka untuk mengidentifikasi perilaku unduhan mencurigakan yang mungkin menunjukkan adanya serangan terkoordinasi atau redistribusi konten yang tidak sah.

Langkah-langkah implementasi

- Gunakan identitas akses CloudFront asal Amazon (OAI) untuk membatasi akses langsung ke objek S3
- Berasosiasi AWS WAF dengan CloudFront atau ALB untuk menyediakan penyaringan layer-7 dan membantu melindungi terhadap serangan DDo S dan permintaan berbahaya.

- Konfigurasikan header HTTP kustom di Cloudfront untuk memverifikasi bahwa lalu lintas masuk berasal dari sumber resmi.

## GAMESEC04-BP03 Menerapkan pembatasan geografis untuk membatasi akses yang tidak sah

Saat pemain meminta konten Anda, Amazon CloudFront menyajikan konten yang diminta dari lokasi tepi terdekat, di mana pun pemain berada. Namun, mungkin ada skenario di mana Anda perlu membatasi bagaimana konten Anda dapat diakses oleh pengguna di bagian dunia tertentu. Misalnya, Anda mungkin memiliki strategi penyebaran game bergulir yang merilis konten secara bertahap country-by-country secara bertahap, atau Anda mungkin harus mematuhi kontrol akses khusus negara.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Anda dapat menggunakan batasan geografis, juga dikenal sebagai pemblokiran geografis, untuk memblokir pemain di lokasi geografis tertentu agar tidak mengakses konten yang Anda distribusikan melalui distribusi. CloudFront Fitur ini memungkinkan Anda membatasi akses ke file yang terkait dengan distribusi dan membatasi akses di tingkat negara. Atau, Anda dapat menggunakan layanan geo-lokasi pihak ketiga untuk membatasi akses ke subset file yang terkait dengan distribusi atau untuk membatasi akses pada perincian yang lebih halus daripada tingkat negara.

Dengan menggunakan batasan CloudFront geografis, Anda dapat mengizinkan pemain Anda untuk hanya mengakses konten Anda jika mereka berada di salah satu negara yang ada dalam daftar negara yang disetujui. Anda juga dapat memblokir pemain Anda dari mengakses konten Anda jika mereka berada di salah satu negara yang berada dalam daftar penolakan negara yang dilarang. Jika permintaan diterima dari lokasi geografis yang diblokir, CloudFront akan mengembalikan kode status HTTP 403 Forbidden ke pemain. Penting untuk dicatat bahwa ini berfungsi dengan baik untuk konten yang tidak sensitif dan tidak boleh digunakan sebagai perlindungan yang berdiri sendiri untuk PII atau artefak game sensitif.

Langkah-langkah implementasi

- Gunakan batasan CloudFront geografis untuk mengizinkan atau menolak akses konten berdasarkan daftar izinkan atau tolak tingkat negara.

- Kembalikan kode status HTTP Terlarang 403 untuk permintaan yang berasal dari lokasi geografis yang diblokir.
- Hindari hanya mengandalkan pembatasan geografis untuk melindungi konten sensitif atau PII

## GAMESEC04-BP04 Membatasi akses ke konten dengan solusi manajemen hak digital (DRM)

Pertimbangkan untuk membatasi akses ke konten game Anda dengan menggunakan alat enkripsi yang kuat seperti solusi manajemen hak digital (DRM). Jenis solusi ini dapat digunakan untuk mengenkripsi konten pribadi Anda dan mendistribusikan kunci dekripsi ke pemain yang berwenang.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Solusi DRM direkomendasikan dalam situasi di mana Anda ingin mengizinkan pemain mengunduh konten game lebih awal, tetapi Anda tidak ingin mereka dapat mengakses atau memutar konten hingga waktu yang telah ditentukan. Misalnya, ini umum terjadi dalam situasi di mana pemain diizinkan untuk memesan di muka game dan mengonfigurasi klien game mereka untuk secara otomatis mulai mengunduh file terenkripsi lebih awal. Strategi ini memverifikasi bahwa game diunduh dan siap dimainkan setelah game dirilis secara resmi. Setelah game dirilis, klien game pemain dapat meminta kunci dekripsi dari solusi backend DRM sehingga dapat mendekripsi file yang diunduh sebelumnya dan mulai memainkan game.

Sistem DRM juga digunakan untuk memblokir distribusi ulang dan manipulasi game yang tidak sah setelah diunduh dan diinstal oleh pemain yang berwenang. Sistem DRM memerlukan integrasi dengan asal untuk bertukar kunci enkripsi dan mengotorisasi pemain untuk mengambil kunci dekripsi. Penyedia DRM komersial menawarkan berbagai solusi dengan fitur dan dukungan untuk perangkat yang berbeda.

Langkah-langkah implementasi

- Gunakan solusi DRM untuk mengenkripsi konten game pribadi dan mendistribusikan kunci dekripsi ke pemain yang berwenang.
- Aktifkan pra-unduhan file terenkripsi untuk game yang dipesan sebelumnya, membuka kunci akses dengan kunci dekripsi pada waktu rilis.
- Integrasikan sistem DRM dengan asal untuk mengelola kunci enkripsi dan memblokir redistribusi atau manipulasi konten yang tidak sah.

# Deteksi

GAMESEC05: Bagaimana Anda memantau dan menganalisis perilaku penggunaan pemain dalam game Anda?

Memantau dan menganalisis perilaku penggunaan pemain sangat penting untuk studio game karena memungkinkan Anda mendeteksi ancaman keamanan, kecurangan, dan bentuk perilaku kasar lainnya yang dapat membahayakan integritas permainan dan keselamatan pemain. Dengan melacak pola seperti tingkat perkembangan yang tidak biasa, transaksi dalam game yang tidak normal, atau perilaku komunikasi yang mencurigakan, Anda dapat mengidentifikasi calon penipu, akun penipuan, atau ancaman terkoordinasi sebelum berdampak signifikan pada pengalaman pemain.

Praktik terbaik

- GAMESEC05-BP01 Menerapkan strategi pengumpulan data yang komprehensif untuk memantau perilaku pemain
- GAMESEC05-BP02 Kumpulkan, simpan, dan analisis log penggunaan pemain untuk mendeteksi perilaku yang tidak pantas

## GAMESEC05-BP01 Menerapkan strategi pengumpulan data yang komprehensif untuk memantau perilaku pemain

Untuk mempertahankan pengalaman pemain yang positif, terapkan strategi pengumpulan dan analisis data yang komprehensif. Menangkap, menyimpan, dan menganalisis data yang relevan memberikan wawasan tentang bagaimana pemain berinteraksi dengan fitur game Anda dan satu sama lain. Pendekatan berbasis data ini dapat memandu pengambilan keputusan, meningkatkan keterlibatan dan retensi pemain, mengoptimalkan strategi monetisasi, dan pada akhirnya meningkatkan pengalaman pemain secara keseluruhan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Menerapkan sistem pengumpulan data untuk menangkap dan mencatat tindakan pemain yang relevan, seperti sesi permainan, kemajuan, pencapaian, pembelian, interaksi dengan elemen

permainan, dan aktivitas sosial. Kumpulkan data sisi server seperti beban server, lalu lintas jaringan, dan log kesalahan untuk memantau kinerja teknis dan mengidentifikasi potensi masalah. Kumpulkan umpan balik pemain melalui survei, forum, tiket dukungan, dan saluran media sosial untuk memahami pengalaman dan preferensi mereka.

Saat menyimpan data game Anda, buat gudang data terpusat atau data lake untuk menyimpan dan mengatur data yang dikumpulkan dan menerapkan jalur pipa untuk pembersihan, transformasi, dan agregasi data guna menyiapkan data untuk analisis yang efisien.

Setelah menyimpan data, analisis untuk mendapatkan wawasan seperti retensi dan churn pemain, strategi monetisasi, dan penggunaan fitur melalui alat visualisasi data.

Langkah-langkah implementasi

- Tangkap dan catat tindakan pemain, metrik sisi server, dan umpan balik untuk memantau interaksi dan kinerja teknis.
- Gunakan gudang data terpusat seperti Amazon Redshift atau S3 data lake untuk menyimpan, membersihkan, mengubah, dan mengatur data game untuk analisis.
- Analisis data yang dikumpulkan dengan alat visualisasi, seperti Amazon Quicksight, untuk mendapatkan wawasan tentang retensi pemain, monetisasi, dan penggunaan fitur.

## GAMESEC05-BP02 Kumpulkan, simpan, dan analisis log penggunaan pemain untuk mendeteksi perilaku yang tidak pantas

Instrumen permainan Anda untuk mengumpulkan log untuk memahami bagaimana pemain menggunakan fitur permainan Anda dan bagaimana mereka berinteraksi dengan pemain lain. Anda kemudian dapat memblokir aktivitas tidak sah yang dapat menurunkan pengalaman pemain.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Kirim peristiwa log terstruktur ke Pipeline Game Analytics, dengan menggunakan solusi logging seperti Amazon CloudWatch Logs atau Amazon OpenSearch Service, atau melalui solusi dari AWS Partner seperti Datadog, Sumo Logic, New Relic, HoneyComb.io, atau Splunk. Struktur log penggunaan pemain ini sehingga mereka dapat digunakan untuk mendeteksi kapan tindakan tertentu oleh pemain perlu diselidiki.

Setelah Anda menangkap data, pertimbangkan untuk menerapkan alat untuk mendeteksi perilaku penggunaan yang tidak pantas. Misalnya, jika game Anda memiliki fitur sosial seperti pesan pemain dalam game, obrolan suara, atau forum online, simpan log dari keterlibatan pemain ini dalam format yang dapat dianalisis untuk tujuan moderasi.

Konfigurasikan fitur obrolan suara game Anda untuk mengekspor rekaman ke Amazon S3 dan gunakan Amazon Transcribe untuk mengonversi pidato audio ke format teks yang dapat disimpan untuk diproses. Atau, Anda dapat melakukan transkripsi streaming waktu nyata dengan mengintegrasikan layanan obrolan suara backend game Anda secara langsung dengan Transcribe API untuk mentranskripsikan audio streaming secara real time. Tim moderasi dapat meninjau konten secara manual, dan setelah konten dalam format standar, Anda juga dapat menggunakan layanan AWS AI/ML untuk melakukan moderasi secara otomatis. Amazon Comprehend dapat digunakan untuk melakukan pemrosesan bahasa alami (NLP) untuk mengungkap informasi dari teks yang tidak terstruktur, yang dapat mengklasifikasikan dan mengatur percakapan ke dalam topik yang relevan dan mengidentifikasi perilaku yang tidak pantas seperti kata-kata kotor.

Langkah-langkah implementasi

- Kumpulkan, simpan, dan analisis log penggunaan pemain.
- Gunakan AWS layanan untuk kecerdasan buatan dan pembelajaran mesin untuk meninjau dan mendapatkan wawasan lebih efisien tentang log penggunaan pemain Anda.

## Perlindungan infrastruktur

Lihat whitepaper Well-Architected Framework untuk praktik terbaik dalam perlindungan infrastruktur untuk keamanan yang berlaku untuk beban kerja game.

GAMESEC06: Bagaimana Anda memantau dan menanggapi ancaman infrastruktur?

Memantau dan menanggapi ancaman infrastruktur sangat penting untuk studio game karena infrastruktur mereka mewakili tulang punggung yang mendukung jutaan pemain bersamaan, memproses transaksi uang nyata, dan menyimpan data pemain yang berharga dan konten game eksklusif. Sangat penting bagi studio game untuk menerapkan sistem pemantauan dan proses respons insiden yang dapat menjaga integritas pengalaman pemain dan operasi bisnis.

Praktik terbaik

- GAMESEC06-BP01 Gunakan alat untuk mendeteksi dan menanggapi ancaman terhadap infrastruktur Anda
- GAMESEC06-BP02 Gunakan kecerdasan buatan dan alat pembelajaran mesin untuk mengotomatiskan aspek strategi perlindungan infrastruktur Anda
- GAMESEC06-BP03 Gunakan wawasan dari log tingkat sistem untuk terus meningkatkan strategi perlindungan infrastruktur Anda

## GAMESEC06-BP01 Gunakan alat untuk mendeteksi dan menanggapi ancaman terhadap infrastruktur Anda

Untuk terus memantau aktivitas berbahaya dan perilaku tidak sah di AWS lingkungan Anda, pertimbangkan untuk menggunakan Amazon GuardDuty. GuardDuty mengidentifikasi ancaman dengan memantau perilaku akun, aktivitas jaringan, dan pola akses data dalam lingkungan Anda. Ini menganalisis peristiwa di berbagai sumber data, seperti log CloudTrail peristiwa, Amazon VPC Flow Logs, dan log DNS untuk potensi ancaman. Dengan mengintegrasikan dengan Amazon CloudWatch Events dan Lambda GuardDuty , peringatan dapat diteruskan secara otomatis ke tim keamanan yang relevan untuk analisis lebih lanjut.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

AWS Security Hub CSPMmemberikan pandangan komprehensif tentang keadaan keamanan Anda AWS dan memeriksa lingkungan Anda terhadap standar industri keamanan dan praktik terbaik. Security Hub CSPM mengumpulkan data keamanan dari berbagai layanan Akun AWS, dan produk mitra pihak ketiga yang didukung serta menganalisis tren keamanan Anda serta mengidentifikasi masalah keamanan prioritas tertinggi. GuardDutyIntegrasi Amazon dengan Security Hub CSPM memungkinkan Anda mengirim temuan dari CSPM Security Hub GuardDuty ke Security Hub. Security Hub CSPM kemudian dapat memasukkan temuan-temuan tersebut dalam analisisnya tentang postur keamanan Anda.

Adalah umum bagi aktor jahat untuk mempekerjakan bot untuk mengambil alih akun dan menipu dalam permainan. WAF Bot Control memberi Anda visibilitas dan kontrol atas lalu lintas bot umum dan meresap yang dapat mengkonsumsi sumber daya berlebih, metrik miring, menyebabkan downtime, atau melakukan aktivitas lain yang tidak diinginkan.

Ransomware adalah kode berbahaya yang dirancang untuk mendapatkan akses tidak sah ke sistem dan kumpulan data dan mengenkripsi data tersebut untuk memblokir akses oleh pemain yang

sah. Setelah ransomware mengunci pemain dari sistem mereka dan mengenkripsi data sensitif mereka, penjahat cyber menuntut uang tebusan sebelum memberikan kunci dekripsi untuk membuka kunci data. Organizations dapat sepenuhnya ditutup oleh peristiwa jahat, menimbulkan biaya yang signifikan dan hilangnya produktivitas bisnis. Lihat Mengamankan AWS Cloud lingkungan Anda dari ransomware untuk praktik terbaik yang dapat Anda terapkan untuk memperkuat kemampuan Anda melawan ransomware sebelum, selama, dan setelah insiden terjadi.

Game Anda dapat memberi pemain kemampuan untuk menghubungi agen dukungan pemain melalui pusat panggilan seperti Amazon Connect atau bot obrolan menggunakan Amazon Lex. Amazon Connect menyediakan dukungan untuk memantau percakapan langsung dan rekaman. Untuk menganalisis interaksi antara pemain dan bot obrolan dukungan pemain yang dibuat dengan Amazon Lex, Anda dapat menyimpan log percakapan dari interaksi ini di Amazon CloudWatch Log yang dapat diekspor ke Amazon S3 dan dianalisis seperti yang dijelaskan sebelumnya.

Terakhir, lakukan latihan pengujian penetrasi sebagai bagian dari strategi perlindungan infrastruktur Anda. Baik Anda melakukan penilaian ini secara internal atau melalui AWS Mitra, patuhi kebijakan dukungan AWS pelanggan untuk pengujian penetrasi.

Langkah-langkah implementasi

- Gunakan Amazon GuardDuty untuk memantau perilaku akun, aktivitas jaringan, dan pola akses data untuk ancaman, dan berintegrasi dengan Security Hub CSPM untuk tampilan keamanan terpadu.
- Terapkan Kontrol AWS WAF Bot untuk membantu mendeteksi dan mengurangi lalu lintas bot yang dapat membahayakan sumber daya dan pengalaman pemain.
- Lakukan latihan pengujian penetrasi secara teratur, mengikuti kebijakan dukungan AWS pelanggan, untuk menilai dan memperkuat postur keamanan Anda.

## GAMESEC06-BP02 Gunakan kecerdasan buatan dan alat pembelajaran mesin untuk mengotomatiskan aspek strategi perlindungan infrastruktur Anda

Amazon Lookout for Metrics menggunakan pembelajaran mesin untuk secara otomatis mendeteksi dan mendiagnosis anomali dalam data bisnis dan operasional Anda serta memantau metrik yang paling penting bagi bisnis Anda dengan kecepatan dan akurasi yang lebih tinggi. Layanan ini juga memudahkan untuk mendiagnosis akar penyebab anomali, seperti penurunan mendadak dalam pendapatan, login, transaksi, atau retensi. Tidak mengharuskan pengembang game untuk memiliki

pengalaman ML untuk mengatur dan dapat terhubung ke sumber data populer termasuk Amazon S3, Amazon, Amazon RDS, CloudWatch Amazon Redshift, serta banyak aplikasi SaaS. Misalnya, Anda dapat mengintegrasikan Amazon Lookout for Metrics dengan Pipeline Game Analytics dan sumber data lainnya untuk mulai menganalisis perilaku guna mendeteksi anomali.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Atau, Anda dapat memilih untuk membuat, melatih, dan menghosting model pembelajaran mesin khusus menggunakan Amazon SageMaker AI AI untuk mengatasi kasus penggunaan seperti moderasi konten, deteksi toksisitas, deteksi cheat, deteksi penipuan, dan banyak lagi.

### Contoh pelanggan

AnyCompany Game menggunakan Amazon Lookout for Metrics untuk secara otomatis mendeteksi pola yang tidak biasa dalam kinerja server, upaya login pemain, atau volume transaksi yang dapat menunjukkan ancaman dari aktor jahat. Selain itu, mereka telah menggunakan Amazon SageMaker AI untuk mengembangkan model pembelajaran mesin khusus yang terus menganalisis pola lalu lintas jaringan dan perilaku pemain untuk membantu mengidentifikasi ancaman terkoordinasi, seperti jaringan bot yang mencoba mengeksploitasi ekonomi virtual mereka.

Pendekatan otomatis ini memungkinkan tim keamanan mereka untuk fokus menyelidiki dan menanggapi ancaman asli daripada memantau ribuan metrik secara manual, sambil memastikan bahwa pola ancaman yang muncul terdeteksi dan ditangani sebelum mereka dapat secara signifikan memengaruhi ketersediaan game atau keamanan pemain.

### Langkah-langkah implementasi

- Gunakan Amazon Lookout for Metrics untuk membantu mendeteksi dan mendiagnosis anomali secara otomatis dalam data bisnis dan operasional utama
- Integrasikan Amazon Lookout for Metrics dengan sumber data seperti Pipeline Game Analytics, Amazon S3, CloudWatch atau untuk memantau metrik seperti pendapatan, login, dan retensi.
- Gunakan Amazon SageMaker AI untuk membuat, melatih, dan menghosting model pembelajaran mesin khusus untuk kasus penggunaan lanjutan seperti deteksi cheat, pencegahan penipuan, dan moderasi konten.

# GAMESEC06-BP03 Gunakan wawasan dari log tingkat sistem untuk terus meningkatkan strategi perlindungan infrastruktur Anda

Tangkap dan simpan log tingkat sistem dari layanan yang relevan, seperti log akses server S3, log akses, dan log CloudFront aksesALB. Log ini dapat disimpan dalam bucket S3 di akun Anda dan berguna untuk mengaitkan informasi penggunaan pemain Anda dari dalam game dengan informasi tingkat sistem termasuk detail koneksi seperti alamat IP, header permintaan, dan manipulasi dan pemfilteran permintaan yang relevan yang mungkin telah Anda konfigurasikan dalam backend game Anda. Anda dapat mengirim log ini ke solusi logging yang sama yang disebutkan sebelumnya, dan Anda dapat menganalisisnya menggunakan kueri SQL dengan Amazon Athena tanpa mengharuskan log dipindahkan dari Amazon S3.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Access Analyzer for S3 adalah fitur yang memantau kebijakan akses bucket Anda, memastikan bahwa kebijakan tersebut hanya menyediakan akses yang dimaksudkan ke sumber daya Amazon S3 Anda. Access Analyzer for S3 mengevaluasi kebijakan akses bucket Anda dan memungkinkan Anda menemukan dan memperbaiki bucket dengan cepat dengan akses yang berpotensi tidak diinginkan.

Langkah-langkah implementasi

- Gunakan AWS layanan untuk deteksi ancaman dan respons insiden untuk mengotomatiskan aspek strategi perlindungan infrastruktur Anda.
- Dapatkan wawasan tentang perlindungan infrastruktur Anda melalui log tingkat sistem dan AWS layanan untuk kecerdasan buatan dan pembelajaran mesin.

## Perlindungan data

Saat mengembangkan dan merancang game Anda, pertimbangkan jenis data apa yang dikumpulkan studio Anda dan bagaimana Anda memutuskan untuk melindunginya. Topik untuk dijelajahi dalam aspek keamanan ini meliputi:

- Bagaimana Anda memilih untuk mengidentifikasi dan mengklasifikasikan data Anda
- Bagaimana Anda melindungi data saat istirahat
- Bagaimana Anda melindungi data dalam perjalanan

Tidak ada praktik terbaik perlindungan data khusus untuk Game Lens. Lihat whitepaper Well-Architected Framework untuk praktik terbaik dalam perlindungan data untuk keamanan.

# Respons insiden

GAMESEC07: Bagaimana Anda mendefinisikan dan menegakkan kebijakan untuk menanggapi pelanggaran pemain dan perilaku kasar?

Pelanggaran pemain dan perilaku kasar dapat secara signifikan memengaruhi pengalaman pemain Anda. Perilaku pemain yang buruk dapat mengusir pemain yang sah, yang menyebabkan penurunan retensi pemain, berkurangnya pendapatan dari pembelian dalam game, dan ulasan negatif yang dapat merusak reputasi game dan penjualan di masa depan.

Tentukan kebijakan yang mempromosikan tindakan positif di antara pemain Anda dan tentukan bagaimana Anda akan menegakkan kebijakan ini.

Praktik terbaik

- GAMESEC07-BP01 Menerapkan rencana respons insiden untuk menangani aktor jahat dan perilaku kasar
- GAMESEC07-BP02 Melarang akun yang terkait dengan aktor jahat

## GAMESEC07-BP01 Menerapkan rencana respons insiden untuk menangani aktor jahat dan perilaku kasar

Buat rencana tindakan untuk menanggapi aktor jahat dan perilaku kasar dalam permainan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Pertimbangkan faktor-faktor seperti kapan harus menangguhkan sementara atau melarang pemain secara permanen dan berapa lama untuk menonaktifkan kredensi untuk pemain yang ditangguhkan sementara.

Contoh pelanggan

AnyCompany Game menciptakan sistem respons insiden berjenjang di mana pelanggaran kecil seperti pesan obrolan yang tidak pantas mengakibatkan penangguhan akun 24 jam otomatis, sementara pelanggaran yang lebih parah seperti kecurangan atau pelecehan memicu penangguhan 7 hari langsung dengan tinjauan wajib oleh moderator manusia.

Selain itu, AnyCompany Games menetapkan prosedur eskalasi di mana pelanggar berulang menghadapi suspensi yang semakin lama. Mereka menciptakan proses banding yang memungkinkan pemain yang ditandai secara salah untuk menentang tindakan otomatis sambil menjaga keamanan melalui persyaratan verifikasi identitas.

## GAMESEC07-BP02 Melarang akun yang terkait dengan aktor jahat

Jika dibiarkan tanpa tanggung-tanggung, perilaku kasar dalam permainan dapat terus berdampak negatif pada pengalaman bermain game bagi orang lain dan harus dikurangi sesegera mungkin. Menerapkan proses untuk memberlakukan larangan atau bentuk pembatasan lain pada aktor jahat yang dikonfirmasi melanggar ketentuan layanan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Biasanya, aturan dan proses evaluasi untuk menentukan keadaan untuk memberlakukan jenis pembatasan ini akan ditentukan oleh personel seperti tim komunitas pemain atau tim kepercayaan dan keselamatan dalam organisasi Anda. Setelah Anda menandai aktor jahat, jalankan alur kerja yang telah ditentukan sebelumnya untuk bertindak pada pemain yang diidentifikasi.

Misalnya, AWS Step Functionsdan AWS Lambdafungsi dapat digunakan untuk menjalankan alur kerja otomatis yang menerima sejumlah akun pemain sebagai input. Alur kerja kemudian memperbarui entri dalam tabel Amazon DynamoDB yang disebut Bans, yang dapat mencakup detail tentang akun pemain, alasan larangan, dan durasi.

Bergantung pada desain permainan dan sistem manajemen akun Anda dan jenis penyalahgunaan yang Anda temui dari aktor jahat, pertahankan sistem larangan catatan yang terpisah dari sistem manajemen akun Anda. Anda mungkin tidak ingin mematikan akun pemain dari sistem manajemen akun Anda, memilih untuk hanya mematikan kemampuan mereka untuk memainkan game Anda. Ini dapat berguna dalam situasi di mana kredensi akun pemain digunakan untuk mengakses beberapa game dengan persyaratan layanan atau kebijakan yang berbeda.

Langkah-langkah implementasi

- Mendefinisikan dan menegakkan kebijakan untuk menanggapi perilaku kasar dari aktor jahat.

- Gunakan AWS layanan untuk mengotomatiskan tanggapan Anda terhadap aktor jahat.

Sumber daya

- AWS Panduan Teknis Respons Insiden Keamanan
- AWS Machine Learning Blog: Mendeteksi pengguna nyata dan langsung dan mencegah aktor jahat menggunakan Amazon Rekognition Face Liveness
- AWS Solusi untuk Game: Kesehatan Masyarakat

# Keamanan aplikasi

GAMESEC08: Bagaimana Anda mengamankan pipa Anda? CI/CD

CI/CD Pipa pengembangan game biasanya terdiri dari server dan penyimpanan kontrol sumber yang sangat tersedia, sumber daya komputasi untuk menjalankan build Anda, dan perangkat lunak untuk melakukan pengujian otomatis, bersama dengan konektivitas jaringan yang tepat dari mesin pengembangan Anda. Mengamankan CI/CD pipeline Anda penting untuk melindungi informasi sensitif, menjaga integritas kode, dan mempertahankan rilis tepercaya. Menanamkan tata kelola dan pagar pembatas memungkinkan kelincahan pengembang sambil mempertahankan praktik keamanan yang baik.

Karena game sering menangani pemrosesan pembayaran, menyimpan informasi pribadi, dan mempertahankan ekonomi virtual yang bernilai uang sungguhan, pelanggaran keamanan dalam proses pengembangan dapat mengakibatkan kerugian finansial yang signifikan, hukuman peraturan, dan hilangnya kepercayaan pemain.

Dengan mengintegrasikan pengamanan, organisasi mempertahankan visibilitas dan kontrol atas proses pengiriman perangkat lunak, memungkinkan respons insiden yang cepat dan mempromosikan budaya praktik pengkodean yang aman.

Praktik terbaik

- GAMESEC08-BP01 Menerapkan keamanan pada setiap tahap pipa CI/CD

## GAMESEC08-BP01 Menerapkan keamanan pada setiap tahap pipa CI/CD

Pagar pembatas seperti kontrol akses, pemisahan tugas, dan jalur audit memberikan perlindungan terhadap akses yang tidak sah atau aktivitas berbahaya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Orang-orang, proses, dan teknologi Anda harus mengamankan pipa juga. Orang-orang yang paling dekat dengan kode harus membangun praktik pengkodean yang aman dan memastikan mereka mengikutinya. Ulangi proses Anda terus menerus untuk memverifikasi bahwa ada konsistensi dalam tingkat keamanan di seluruh pipeline. Terakhir, terapkan teknologi untuk memverifikasi bahwa praktik dan proses terbaik tidak dilewati.

Contoh pelanggan

AnyCompany Game menerapkan kontrol akses berbasis peran di mana hanya pengembang senior yang dapat menyetujui perubahan pada kode sistem anti-cheat mereka, sementara memerlukan tinjauan kode wajib dari anggota tim keamanan untuk komponen yang menangani data pembayaran pemain.

CI/CD Pipeline mereka secara otomatis menjalankan pemeriksaan validasi model ancaman, memastikan bahwa fitur baru seperti pasar perdagangan pemain diuji terhadap vektor serangan yang diidentifikasi sebelumnya seperti eksploitasi duplikasi item atau upaya transaksi penipuan.

Langkah-langkah implementasi

- Berikan izin pengguna berdasarkan prinsip hak istimewa paling sedikit.
- Gunakan AWS CloudTrail untuk mengaudit panggilan API yang dilakukan di seluruh layanan yang digunakan dalam pipeline.
- Gunakan kait pra-komit untuk memverifikasi bahwa kode mengikuti praktik umum dan kebijakan perusahaan.

# Mengotomatiskan keamanan

GAMESEC09: Bagaimana Anda mengotomatiskan keamanan di dalam pipa Anda? CI/CD

Integrasikan langkah-langkah keamanan dalam CI/CD pipeline Anda untuk mempertahankan postur keamanan yang kuat sepanjang siklus hidup pengembangan. Proses ini menawarkan banyak manfaat yang sama seperti memiliki keamanan pipa Anda. Memiliki CI/CD pipeline yang aman mengurangi kemungkinan peristiwa keamanan yang berpotensi menunda timeline pengembangan game.

Menyediakan keamanan dalam CI/CD pipeline Anda melibatkan penerapan praktik dan alat keamanan terbaik di setiap tahap siklus pengembangan. Memiliki keamanan dalam pipa juga memungkinkan Anda untuk juga mengurangi waktu tinjauan keamanan.

Mengotomatiskan keamanan dalam CI/CD pipeline Anda sangat penting untuk memverifikasi bahwa kontrol keamanan diterapkan dan diuji secara konsisten dengan setiap perubahan kode. Menerapkan alat dan otomatisasi yang tepat dapat memberikan permainan yang aman dan terjamin.

Praktik terbaik

- GAMESEC09-BP01 Integrasikan perkakas dan otomatisasi untuk mengurangi waktu rata-rata tinjauan keamanan

## GAMESEC09-BP01 Integrasikan perkakas dan otomatisasi untuk mengurangi waktu rata-rata tinjauan keamanan

Untuk mengidentifikasi kerentanan keamanan, organisasi dapat menggunakan berbagai alat dan layanan yang berbeda seperti Pengujian Keamanan Aplikasi Statis (SAST) dan Pengujian Keamanan Aplikasi Dinamis (DAST). SAST adalah cara untuk meninjau kode sumber dan menentukan kerentanan keamanan. DAST adalah cara kotak hitam untuk menguji kode Anda yang menguji aplikasi Anda tanpa melihat kode sumber.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Alat lain yang dapat digunakan organisasi adalah Analisis Komposisi Perangkat Lunak (SCA), yang menilai keamanan dependensi pihak ketiga atau open source Anda. Untuk pendekatan yang lebih manual, tinjauan kode aman dapat diimplementasikan di seluruh pipeline.

Contoh pelanggan

AnyCompany Game menggunakan alat SAST untuk secara otomatis menandai potensi kelemahan keamanan selama proses pengembangan. Mereka juga menggunakan alat DAST untuk

mensimulasikan ancaman terhadap menjalankan game build untuk memvalidasi bahwa kontrol keamanan berfungsi sebagaimana dimaksud. Selain itu, AnyCompany Game mengintegrasikan alat pemindaian ketergantungan ke dalam proses pengembangannya untuk secara otomatis mengidentifikasi kerentanan yang diketahui di perpustakaan pihak ketiga dan mesin game.

Langkah-langkah implementasi

- Gunakan Amazon CodeGuru sebagai alat SAST.
- Gunakan alat sumber terbuka seperti OWASP Dependency Check,, atau. SonarQube OWASPZap

Sumber daya

- Keamanan untuk Pengembang

## Pemodelan ancaman

GAMESEC10: Bagaimana Anda mengintegrasikan pemodelan ancaman ke dalam siklus pengembangan aplikasi organisasi Anda?

Pemodelan ancaman adalah proses mengidentifikasi dan memprioritaskan potensi ancaman terhadap aplikasi Anda dan menentukan solusi yang dapat digunakan untuk menguranginya. Praktik ini menjadi semakin penting karena game telah berkembang menjadi sistem yang kompleks dan terhubung yang menangani data pengguna yang sensitif dan transaksi uang nyata.

Integrasikan pemodelan ancaman sebagai latihan berkelanjutan untuk mendukung keamanan permainan, tidak hanya pada fase desain awal tetapi karena permainan terus tumbuh dan berkembang.

Praktik terbaik

- GAMESEC10-BP01 Tentukan kapan dan bagaimana menyelesaikan latihan pemodelan ancaman di seluruh siklus pengembangan aplikasi

# GAMESEC10-BP01 Tentukan kapan dan bagaimana menyelesaikan latihan pemodelan ancaman di seluruh siklus pengembangan aplikasi

Tidak ada satu cara terbaik untuk mendekati pemodelan ancaman. Detail kapan dan bagaimana melakukan ini akan bervariasi berdasarkan kebutuhan unik studio game Anda. Misalnya, tergantung pada ukuran studio Anda, Anda mungkin memiliki anggota tim yang terlibat dalam satu atau beberapa aspek proses pemodelan ancaman.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Blog AWS Keamanan memberikan gambaran umum untuk pertimbangan yang perlu diingat ketika merancang strategi Anda untuk pemodelan ancaman, seperti:

- Manakah dari anggota tim dan persona Anda yang harus terlibat dalam pemodelan ancaman
- Cara menentukan alat alur kerja yang sesuai untuk digunakan
- Cara menentukan kepemilikan berbagai aspek pemodelan ancaman
- Cara mengidentifikasi dan mengevaluasi kontrol keamanan yang akan digunakan dalam desain beban kerja Anda

### Contoh pelanggan

AnyCompany Game dimulai dengan membuat katalog aset berharga seperti data pemain, kode game dan algoritme, mata uang dalam game, konten yang dibuat pengguna, dan kekayaan intelektual seperti konten yang belum dirilis atau mesin berpemilik. Mereka mempertimbangkan berbagai jenis aktor jahat potensial seperti penipu yang mencari keuntungan yang tidak adil, aktor jahat yang mencoba mencuri data pribadi atau keuangan, dan pengguna jahat yang mencoba mengganggu permainan.

Sepanjang proses pengembangan, AnyCompany Games menggunakan model ancaman untuk memandu praktik pengkodean yang aman dan memengaruhi strategi pengujian untuk fokus pada area berisiko tinggi. Sebelum peluncuran game, mereka melakukan tinjauan pemodelan ancaman komprehensif untuk menilai kesiapan untuk beban pemain yang diantisipasi dan upaya akses yang tidak sah, dan untuk mempersiapkan prosedur respons insiden.

### Langkah-langkah implementasi

- Terapkan pagar pembatas di setiap tahap pipa Anda CI/CD .

- Gunakan otomatisasi dan alat untuk meningkatkan efisiensi ulasan keamanan aplikasi Anda.
- Gunakan pemodelan ancaman sebagai proses untuk meningkatkan keamanan aplikasi Anda.

Sumber daya

- AWS Blog Keamanan: Cara mendekati pemodelan ancaman
- NIST: Panduan untuk Pemodelan Ancaman Sistem Data-Centric
- Pemodelan ancaman dengan cara yang tepat untuk pembangun — AWS Pelatihan mandiri virtual Skill Builder
- Pemodelan ancaman untuk pembangun — AWS Lokakarya

# Sumber daya

Lihat sumber daya berikut untuk mempelajari lebih lanjut tentang praktik terbaik kami terkait keamanan.

Dokumen terkait:

- Skenario Amazon Cognito yang umum
- Menggunakan ditandatangani URLs
- Gunakan alur saluran untuk menghapus konten yang tidak senonoh dan sensitif dari pesan di perpesanan Amazon Chime SDK
- Keamanan di Amazon GameLift
- Mengamankan pengiriman konten menggunakan Amazon CloudFront
- Panduan Respons Keamanan
- AWS Praktik Terbaik untuk Ketahanan DDo S
- Mengamankan AWS Cloud lingkungan Anda dari ransomware

Solusi mitra terkait:

- Datadog
- Logika Sumo
- Splunk
- Honeycomb.io

- New Relic
- AWS Marketplace - Solusi DRM

Materi pelatihan terkait:

- Memulai dengan Amazon Cognito
- Pelatihan mandiri keamanan

# Keandalan

Pilar keandalan mencakup kemampuan sistem untuk pulih dari gangguan infrastruktur atau layanan, memperoleh sumber daya komputasi secara dinamis untuk memenuhi permintaan, dan mengurangi gangguan seperti kesalahan konfigurasi atau masalah jaringan sementara.

Area fokus

- Prinsip desain
- Fondasi
- Arsitektur beban kerja
- Manajemen perubahan
- Manajemen kegagalan
- Sumber daya

## Prinsip desain

Selain prinsip-prinsip desain dalam whitepaper AWS Well-Architected Framework, berikut ini adalah prinsip-prinsip desain yang dapat meningkatkan keandalan di cloud untuk beban kerja game:

- Tetapkan dasar untuk target konkurensi pemain puncak dan skalabilitas sistem yang diperlukan untuk memenuhi proyeksi bisnis: Sebelum meluncurkan game dan selama operasi permainan langsung, kembangkan perkiraan jumlah pemain bersamaan yang diharapkan pada puncak untuk menetapkan sasaran target skalabilitas sistem untuk memenuhi proyeksi ini. Ini membantu menciptakan dasar untuk keandalan game Anda. Tentukan kebijakan penskalaan untuk mengakomodasi perubahan permintaan secara otomatis tanpa memengaruhi ketersediaan dengan memverifikasi bahwa sistem penskalaan Anda mengelola sesi pemain aktif dengan anggun.
- Ukur keandalan Anda dan dampaknya pada pengalaman pemain: Tentukan indikator kinerja utama (KPIs) yang mewakili kesehatan permainan Anda. Pantau dampak perubahan infrastruktur dan fitur game pada keandalan Anda.

# Fondasi

Untuk mencapai keandalan, suatu sistem harus memiliki fondasi dan pemantauan yang terencana dengan baik dengan mekanisme untuk menangani perubahan permintaan atau persyaratan. Sistem harus dirancang untuk mendeteksi kegagalan dan secara otomatis menyembuhkan dirinya sendiri.

Tidak ada dasar praktik terbaik khusus untuk Game Lens. Lihat whitepaper Well-Architected Framework untuk praktik terbaik di fondasi keandalan yang berlaku untuk beban kerja game.

# Arsitektur beban kerja

GAMEREL01: Apakah arsitektur game Anda memanfaatkan ketahanan cloud?

AWS infrastruktur dibangun di sekitar Wilayah dan Zona Ketersediaan. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung menggunakan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Konstruksi ini dapat digunakan untuk merancang beban kerja dengan tujuan keandalan dalam fokus.

Praktik terbaik

- GAMEREL01-BP01 Mendistribusikan infrastruktur game di beberapa Availability Zone dan Region untuk meningkatkan ketahanan

## GAMEREL01-BP01 Mendistribusikan infrastruktur game di beberapa Availability Zone dan Region untuk meningkatkan ketahanan

Untuk meminimalkan dampak kerusakan infrastruktur lokal pada pemain Anda, Anda harus mendistribusikan penyebaran infrastruktur Anda secara seragam di lokasi independen yang cukup untuk dapat menahan gangguan yang tidak terduga sambil tetap memiliki kapasitas yang cukup untuk memenuhi kebutuhan permintaan pemain Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Saat menerapkan infrastruktur game Anda, disarankan untuk mendistribusikan kapasitas Anda secara merata di beberapa Availability Zone di suatu Wilayah sehingga Anda dapat menahan

gangguan pada satu atau beberapa Availability Zone tanpa mengganggu pengalaman pemain. Layanan backend game seperti aplikasi web harus diseimbangkan beban di beberapa Availability Zone atau harus dibangun menggunakan layanan terkelola seperti AWS Lambda dan Amazon API Gateway yang menyediakan ketersediaan tinggi Regional berdasarkan desain. Demikian pula, komponen yang mempertahankan status seperti cache, database, antrian pesan, dan solusi penyimpanan harus dirancang untuk memberikan persistensi data yang tahan lama di beberapa Availability Zone, yang disediakan oleh desain dalam layanan seperti Amazon S3, DynamoDB, dan Amazon SQS, dan dapat dikonfigurasi di layanan lain.

Saat merancang arsitektur hosting server game Anda untuk ketahanan, gunakan armada server game Anda secara seragam di seluruh Availability Zone dalam sebuah Wilayah AWS untuk memaksimalkan akses Anda ke kapasitas komputasi yang tersedia di Wilayah serta mengurangi cakupan dampak gangguan Availability Zone. Misalnya, Anda dapat mengonfigurasi Amazon EC2 Auto Scaling untuk menggunakan Availability Zones. Jika EC2 instance menjadi tidak sehat, EC2 Auto Scaling dapat menggantikan instance, serta meluncurkan instance ke Availability Zone lainnya jika satu atau beberapa Availability Zone menjadi tidak tersedia.

Untuk infrastruktur penting, seperti otentikasi, berikan jumlah minimum instans yang layak yang berjalan di beberapa Availability Zone, dan gunakan penskalaan otomatis untuk menangani peningkatan beban atau toleransi kesalahan jika ada gangguan pada salah satu Availability Zone.

Terapkan infrastruktur game Anda ke beberapa Wilayah untuk memaksimalkan ketersediaan. Fitur pemulihan bencana lintas-regional seperti database global Aurora dan infrastruktur redundan yang dapat menjadi aktif dengan perubahan DNS sederhana yang dikerahkan ke Wilayah sekunder dapat memberikan kontinuitas layanan jika Wilayah primer menjadi terganggu. Meskipun kami mendorong ini untuk layanan backend game Anda untuk mencapai ketersediaan tinggi, rekomendasi ini sangat penting untuk server game Anda.

Misalnya, dalam game multipemain, kapasitas infrastruktur Anda untuk server game cenderung melebihi kebutuhan kapasitas untuk layanan Anda yang lain, karena server game digunakan untuk menyelenggarakan sesi game untuk pemain. Banyak game memilih untuk membelah pemain ke Wilayah permainan logis (seperti AS barat dan timur). Untuk menyederhanakan pengalaman pemain dan membuatnya mudah untuk menggunakan infrastruktur global untuk menyelenggarakan game, pertimbangkan untuk menghapus nama Wilayah game yang menghadap pemain Anda dari Wilayah penyedia cloud yang mendasarinya atau lokasi pusat data yang secara fisik menghosting server game, bersama dengan infrastruktur lain seperti Local Zones atau pusat data Anda sendiri yang menghosting instance server game yang mendukung Wilayah game pemain tersebut.

Saat merancang layanan perjodohan Anda, terapkan arsitektur Multi-wilayah dengan penerapan perangkat lunak terpisah di seluruh Wilayah. Pisahkan penyebaran layanan perjodohan Anda dari armada yang menghosting instance server game Anda sehingga Anda dapat merutekan pemain ke server game di Wilayah terlepas dari penyebaran regional mana dari layanan perjodohan Anda yang menangani permintaan perjodohan.

Rancang logika dalam implementasi perjodohan Anda untuk mendukung Wilayah server game yang memenuhi latensi Anda dan aturan lainnya, dengan kemampuan untuk mundur ke perutean pemain ke Wilayah lain jika armada Anda berkapasitas rendah atau ada gangguan infrastruktur regional lainnya.

Langkah-langkah implementasi

- Mendistribusikan infrastruktur game secara seragam di beberapa Availability Zone untuk memberikan ketersediaan dan ketahanan yang tinggi.
- Terapkan layanan backend game dan komponen stateful menggunakan managed like, Amazon AWS Lambda S3, DynamoDB, dan SQS, atau konfigurasikan load balancing dan durabilitas untuk solusi kustom.
- Menerapkan penyebaran Multi-wilayah untuk layanan dan server game kritis, menggunakan solusi pemulihan bencana seperti database global Aurora dan Wilayah yang menghadap pemain logis yang dipisahkan dari lokasi fisik yang mendasarinya.

Sumber daya

- Stabilitas statis
- Praktik terbaik untuk GameLift antrian sesi game Amazon
- Armada GameLift Multi-Wilayah Amazon
- Database Global Aurora untuk Pemulihan Bencana

# Manajemen perubahan

GAMEREL02: Bagaimana Anda menskalakan game stateful Anda untuk mengakomodasi perubahan permintaan?

Karena permintaan pemain Anda berfluktuasi dari waktu ke waktu, infrastruktur permainan Anda harus dapat menyesuaikan skala untuk menangani persyaratan yang berubah ini. Meskipun sulit untuk memprediksi popularitas game sebelumnya, rancang pendekatan arsitektur yang memungkinkan penambahan dan penghapusan kapasitas infrastruktur untuk mengakomodasi fluktuasi populasi pemain.

Praktik terbaik

- GAMEREL02-BP01 Menerapkan strategi penskalaan yang menggabungkan keadaan sesi permainan pemain aktif
- GAMEREL02-BP02 Mendukung penggunaan beberapa jenis instans untuk game Anda EC2

## GAMEREL02-BP01 Menerapkan strategi penskalaan yang menggabungkan keadaan sesi permainan pemain aktif

Terapkan solusi untuk menskalakan infrastruktur game Anda secara otomatis dengan cara yang menggabungkan sifat stateful dari sesi pemain Anda yang terhubung secara aktif dan menangani aktivitas penskalaan dengan anggun tanpa mengganggu gameplay.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Salah satu keuntungan mengembangkan game di cloud adalah elastisitas yang dapat dicapai dengan menskalakan infrastruktur server secara otomatis sesuai kebutuhan untuk memenuhi permintaan. Meskipun game stateless atau asinkron dan layanan backend dapat diskalakan secara dinamis menggunakan kebijakan Auto EC2 Scaling Amazon, penskalaan otomatis EKS, atau teknik serupa yang biasanya diadopsi untuk aplikasi web yang dapat diskalakan, pengembang game biasanya memerlukan pendekatan yang lebih disesuaikan untuk penskalaan game stateful atau sinkron untuk membantu memblokir gangguan pada sesi pemain aktif.

Langkah-langkah implementasi

- Untuk game stateful, buat metrik khusus yang dapat digunakan untuk memantau status sesi pemain Anda dan kapasitas server game yang tersedia, yang dapat dilaporkan ke Amazon CloudWatch sebagai metrik khusus. Fitur latihan dengan pemantauan aplikasi, seperti CloudWatch Synthetics, memeriksa permainan untuk gangguan fungsi yang pemantauan kesehatan naik turun sederhana mungkin tidak terdeteksi.

- Menggunakan metrik khusus, terapkan perangkat lunak penskalaan server game, misalnya, sebagai aplikasi tanpa server yang menggunakan AWS Lambda fungsi atau AWS Fargate, untuk mengelola armada instance server game khusus dengan menggunakan AWS SDK untuk melakukan panggilan API guna memperbarui pengaturan kapasitas minimum, maksimum, dan yang diinginkan untuk grup Auto Scaling EC2 yang menghosting build server game Anda.
- Gunakan Amazon GameLift untuk meng-host server game Anda dan gunakan kemampuan penskalaan otomatis server out-of-the-box game untuk mengelola proses penskalaan ini untuk Anda.

Kemampuan penskalaan otomatis Amazon GameLift mengetahui sesi pemain aktif dan dapat dikonfigurasi untuk memblokir penghentian atau penskalaan instance server game yang secara aktif menghosting pemain. Untuk informasi selengkapnya, lihat Memantau GameLift Server Amazon dengan Amazon CloudWatch.

## GAMEREL02-BP02 Mendukung penggunaan beberapa jenis instans untuk game Anda EC2

Saat menghosting game Anda menggunakan EC2 instance, atau jika Anda menggunakan container yang dihosting pada EC2 instance Akun AWS, gunakan beberapa jenis instans dalam strategi hosting Anda. Dengan menggunakan beberapa jenis instans, Anda dapat meningkatkan jumlah opsi komputasi yang dapat digunakan saat game Anda ditingkatkan untuk menambahkan lebih banyak server untuk mendukung pertumbuhan pemain, yang meningkatkan keandalan jika jenis instans pilihan Anda tidak tersedia.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Saat menghosting game Anda menggunakan Instans Spot, gunakan beberapa jenis instans karena ketersediaan Instans Spot berfluktuasi berdasarkan permintaan pelanggan.

Uji game Anda pada beberapa jenis instans untuk memenuhi persyaratan biaya dan kinerja Anda dan tentukan peringkat jenis instans yang diprioritaskan. Amazon EC2 Auto Scaling mendukung penggunaan beberapa jenis dan ukuran instans serta menetapkan bobot untuk setiap jenis instans dalam konfigurasi sehingga Anda dapat menerapkan peringkat opsi komputasi yang diprioritaskan.

Saat menghosting game Anda menggunakan hosting GameLift terkelola Amazon, tentukan jenis instance apa yang dibutuhkan game Anda dan cara menjalankan proses server game di dalamnya

(menggunakan konfigurasi runtime). Saat memilih sumber daya untuk armada, pertimbangkan beberapa faktor, termasuk sistem operasi game, jenis instans (perangkat keras komputasi), dan apakah akan menggunakan Instans Sesuai Permintaan, Instans Spot, atau keduanya. Biaya hosting dengan Amazon GameLift terutama tergantung pada jenis instance yang Anda gunakan. Untuk informasi selengkapnya, lihat Memilih sumber daya komputasi untuk armada terkelola.

Langkah-langkah implementasi

- Gunakan beberapa jenis EC2 instans untuk meningkatkan keandalan dan opsi penskalaan saat menghosting game Anda di EC2 atau kontainer.
- Konfigurasikan EC2 Auto Scaling Amazon atau GameLift armada dengan jenis dan bobot instans yang diprioritaskan untuk mengoptimalkan biaya dan kinerja.
- Uji game Anda pada berbagai jenis instans untuk memverifikasi bahwa kinerja memenuhi persyaratan dan sesuaikan strategi hosting Anda .

## Manajemen kegagalan

GAMEREL03: Bagaimana Anda mempertahankan status game selama gangguan infrastruktur?

Karena infrastruktur game Anda mengalami berbagai acara operasional dari waktu ke waktu, arsitektur game Anda harus dirancang untuk menjaga kesinambungan pengalaman pemain dan melestarikan status game selama acara infrastruktur. Untuk menangani peristiwa ini, terapkan pemantauan, shutdown yang anggun, dan mekanisme ketekunan status untuk memverifikasi pengalaman gameplay yang lancar bagi pemain Anda.

Praktik terbaik

- GAMEREL03-BP01 Memantau gangguan server game, dan gunakan data untuk meningkatkan arsitektur hosting untuk mencapai tujuan keandalan
- GAMEREL03-BP02 Menerapkan kopling longgar fitur game untuk menangani kegagalan dengan dampak minimal pada pengalaman pemain
- GAMEREL03-BP03 Memantau peristiwa infrastruktur dari waktu ke waktu untuk mengukur dampak pada perilaku pemain

# GAMEREL03-BP01 Memantau gangguan server game, dan gunakan data untuk meningkatkan arsitektur hosting untuk mencapai tujuan keandalan

Pantau metrik server game dan dampak kegagalan atau penurunan kinerja, seperti peningkatan latensi di bawah beban, pada perilaku pemain dari waktu ke waktu sehingga Anda dapat menyesuaikan strategi hosting server game Anda untuk memenuhi persyaratan keandalan game Anda. Infrastruktur server game yang akan terdegradasi harus segera dihapus dari layanan jika berdampak pada pemain atau diganti secara proaktif ketika tidak ada sesi pemain aktif yang dihosting di server.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Untuk skenario di mana game di-host sebagai REST APIs, keandalan sistem dapat dikelola seperti arsitektur aplikasi web tradisional, di mana lalu lintas dapat dimuat seimbang di beberapa server secara terdistribusi untuk mengurangi risiko kegagalan server.

Untuk gameplay sinkron real-time, sesi permainan biasanya di-host pada proses server game yang berjalan pada mesin virtual, atau instance server game, karena status gameplay perlu dipertahankan dengan cara yang berkinerja dan direplikasi ke klien game yang terhubung. Implementasi ini berarti bahwa pengalaman pemain digabungkan erat dengan kinerja dan keandalan proses server game yang menjadi tuan rumah sesi permainan mereka. Jenis arsitektur ini membuat pengelolaan keandalan server game lebih kompleks daripada pendekatan tradisional.

Untuk mengurangi dampak kegagalan server game, konfigurasikan game Anda untuk terus melakukan pembaruan asinkron status game pemain ke cache atau database yang sangat tersedia seperti Amazon ( ElastiCache Redis OSS) atau Amazon MemoryDB. Jika terjadi kegagalan server, status game terakhir pemain yang disimpan dapat diambil dari penyimpanan data eksternal, dan sesi mereka dapat dipulihkan pada instance server game baru.

Namun, pendekatan ini menambah biaya dan kompleksitas tambahan untuk mengelola keadaan eksternal ini, dan mungkin tidak cocok untuk game yang serba cepat atau kompetitif di mana perubahan status begitu sering terjadi dan terjadi pada skala yang signifikan sehingga memperkenalkan bahkan penyimpanan data cache dalam memori yang berkinerja tinggi akan mengakibatkan kelambatan replikasi yang terlalu signifikan untuk berguna untuk memulihkan sesi. Untuk permainan seperti ini, pendekatan optimal adalah menerima hilangnya server dan mengirim pemain kembali ke lobi permainan untuk menemukan sesi lain atau Anda dapat secara otomatis mengarahkan mereka ke sesi permainan lain.

Tangkap sebanyak mungkin data log yang berguna tentang apa yang menyebabkan gangguan server sehingga Anda dapat menyelidiki masalah ini nanti. Amazon GameLift memberikan panduan untuk men-debug masalah armada dan menyediakan kemampuan untuk mengakses instans GameLift armada Amazon dari jarak jauh.

Langkah-langkah implementasi

- Pantau metrik server game untuk penurunan kinerja, dan hapus atau ganti server yang terdegradasi seperlunya untuk menjaga keandalan.
- Gunakan Amazon ElastiCache atau MemoryDB untuk pembaruan status game asinkron untuk mengaktifkan pemulihan sesi setelah kegagalan server bila memungkinkan.
- Tangkap data log terperinci tentang gangguan server untuk penyelidikan dan debugging, memanfaatkan alat seperti Amazon GameLift untuk pemantauan armada dan akses jarak jauh.

## GAMEREL03-BP02 Menerapkan kopling longgar fitur game untuk menangani kegagalan dengan dampak minimal pada pengalaman pemain

Komponen decoupling mengacu pada konsep merancang komponen server sehingga mereka dapat beroperasi semandiri mungkin. Beberapa aspek permainan sulit dipisahkan karena data harus diperbarui semaksimal mungkin untuk memberikan pengalaman dalam game yang baik bagi para pemain. Namun, banyak komponen dan tugas game dapat dipisahkan. Misalnya, papan peringkat dan layanan statistik tidak penting untuk pengalaman bermain game, dan membaca dan menulis ke layanan ini dapat dilakukan secara asinkron dari game.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Terapkan degradasi anggun untuk fitur dalam game Anda yang dapat dinonaktifkan secara otomatis atau oleh administrator jika masalah terdeteksi, serta konfigurasikan layanan upstream yang bergantung pada fitur untuk dapat menangani kegagalan dengan anggun. Misalnya, jika data pemain tertentu tidak dimuat dengan benar dalam klien game Anda, Anda harus mempertimbangkan apakah data ini penting untuk pengalaman bermain game. Jika tidak, konfigurasikan klien game untuk menangani kegagalan ini dengan anggun tanpa mengganggu pengalaman pemain, memilih untuk mencoba lagi mengambil data ini nanti ketika pemain mengunjungi kembali layar.

Gunakan logika seperti batas waktu, percobaan ulang, dan backoff untuk menangani kesalahan dan kegagalan. Batas waktu menjaga sistem agar tidak menggantung untuk waktu yang terlalu lama. Percobaan ulang dapat memberikan ketersediaan kesalahan sementara dan acak yang tinggi.

Tentukan komponen non-kritis yang dapat digabungkan secara longgar ke komponen kritis. Kopling longgar memungkinkan sistem menjadi lebih tangguh karena kegagalan dalam satu komponen tidak mengalir ke komponen lain. Ketika fitur game tidak memerlukan koneksi stateful ke server game atau backend Anda, Anda harus menerapkan protokol stateless untuk menskalakan secara dinamis dan pulih dari kegagalan sementara. Kembangkan komponen non-kritis Anda yang dapat digabungkan secara longgar dengan protokol stateless menggunakan API. HTTP/JSON Terapkan panggilan jaringan dari klien game agar tidak sinkron dan tidak memblokir untuk meminimalkan dampak bagi pemain dari fitur game yang berkinerja lambat atau layanan dependen lainnya.

Untuk lebih meningkatkan ketahanan melalui kopling longgar, gunakan layanan pesan seperti antrian, streaming, atau sistem berbasis topik antar komponen yang dapat ditangani secara asinkron. Model ini cocok untuk interaksi yang tidak memerlukan tanggapan segera atau di mana pengakuan bahwa permintaan telah terdaftar sudah cukup. Solusi ini melibatkan satu komponen yang menghasilkan peristiwa dan komponen lain yang mengkonsumsinya. Kedua komponen tidak akan terintegrasi melalui point-to-point interaksi langsung tetapi melalui perantara seperti penyimpanan yang tahan lama atau lapisan antrian. Ini juga membantu meningkatkan keandalan sistem dengan melestarikan pesan saat pemrosesan gagal.

Teliti dan pilih mekanisme pesan yang sesuai, karena berbagai layanan pesan memiliki karakteristik yang berbeda, seperti mekanisme pemesanan dan pengiriman. Desain operasi menjadi idempoten sehingga sistem pesan yang dipilih mengirimkan pesan setidaknya sekali. Sebagai contoh, pertimbangkan kasus penggunaan game yang khas di mana game Anda perlu melacak waktu bermain pemain, statistik, atau data relevan lainnya yang dapat menyebabkan kasus penggunaan throughput tulis yang tinggi pada saat konkurensi pemain puncak.

Untuk menerapkan arsitektur yang andal, pertimbangkan apakah use case membutuhkan read-after-write konsistensi seperti yang dirasakan oleh pemain. Biasanya, skenario seperti ini cocok untuk pemrosesan asinkron dan dapat dicapai dengan menerapkan pola antrian tulis di mana permintaan dimasukkan ke dalam antrian pesan yang dapat diskalakan dan tahan lama seperti Amazon SQS dan dapat dimasukkan ke dalam basis data backend Anda dalam batch menggunakan layanan konsumen, seperti fungsi Lambda. Pendekatan ini lebih dapat diandalkan daripada komunikasi sinkron antara beberapa komponen terdistribusi termasuk klien game pemain, web backend dan server aplikasi Anda, dan sistem database internal Anda. Ini juga mengurangi biaya karena basis data backend tidak perlu diskalakan untuk memenuhi throughput penulisan puncak karena

pemrosesan konsumen dari antrian tulis dapat digunakan untuk memperlambat laju konsumsi ini sesuai kebutuhan.

Langkah-langkah implementasi

- Pisahkan komponen non-kritis seperti papan peringkat dan layanan statistik dari fitur gameplay penting untuk memungkinkan operasi asinkron dan meningkatkan ketahanan.
- Terapkan degradasi anggun untuk fitur non-kritis dengan logika untuk batas waktu, percobaan ulang, dan backoff, dan verifikasi bahwa klien game menangani kegagalan tanpa mengganggu pengalaman pemain.
- Gunakan sistem pesan seperti Amazon SQS untuk komunikasi asinkron antar komponen, memungkinkan pemrosesan kasus penggunaan throughput tinggi yang dapat diskalakan, tahan lama, dan andal.

Sumber daya

- Bangun beban kerja yang sangat skalabel dan andal menggunakan arsitektur microservice
- Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa server AWS
- Memahami pesan asinkron untuk layanan mikro
- Pengantar Pola Pengembangan Game yang Dapat Diskalakan AWS
- Menerapkan Degradasi Anggun

## GAMEREL03-BP03 Memantau peristiwa infrastruktur dari waktu ke waktu untuk mengukur dampak pada perilaku pemain

Pantau proses server game Anda, metrik instance server game, dan metrik pengalaman game untuk menentukan akar penyebab masalah. Selain memantau CPU dan memori, Anda juga dapat mengatur pemantauan untuk metrik jaringan yang terkait dengan keterbatasan jaringan EC2 instance untuk mengingatkan Anda tentang masalah seperti melebihi bandwidth, packets-per-second, atau masalah tingkat jaringan lainnya yang mungkin menunjukkan sumber daya server Anda sedang disediakan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Gunakan CloudWatch Synthetics untuk memeriksa fungsionalitas aplikasi jalur kritis untuk pengalaman pemain, seperti tidak dapat masuk atau masalah yang memengaruhi layanan lainnya. Untuk server game yang dihosting menggunakan Amazon GameLift, pertimbangkan untuk memantau metrik seperti:

- GameServerInterruptions dan InstanceInterruptions, yang dapat membantu memahami bagaimana batasan dalam ketersediaan instans Spot memengaruhi server game Anda yang digunakan menggunakan Spot.
- ServerProcessAbnormalTerminations, yang dapat digunakan untuk mendeteksi penghentian abnormal dalam proses server game Anda.

Disarankan untuk mempertahankan data metrik historis keandalan server game Anda. Gunakan data historis ini untuk tujuan pelaporan dan gabungkan dengan kumpulan data lain untuk mengungkap tren potensial dan menilai dampak pada perilaku pemain dari waktu ke waktu yang mungkin disebabkan oleh masalah server game.

Amazon CloudWatch tidak mempertahankan metrik tanpa batas waktu, dan resolusi penyimpanan metrik meningkat dari waktu ke waktu, jadi pertimbangkan untuk mengekspor metrik ini ke penyimpanan jangka panjang yang hemat biaya seperti Amazon S3. Anda dapat mengonfigurasi Aliran CloudWatch Metrik untuk mengirimkan metrik secara otomatis dari CloudWatchWilayah ke bucket S3 Anda sendiri, di mana metrik tersebut dapat disimpan dalam jangka panjang di tingkat penyimpanan seperti S3 Intelligent-Tiering dan akhirnya diarsipkan menggunakan Amazon Glacier. Dengan menempatkan metrik Anda di Amazon S3, metrik tersebut tersedia untuk digabungkan dengan kumpulan data lain di danau data Anda untuk kueri interaktif dengan Amazon Athena.

Langkah-langkah implementasi

- Pantau metrik server, instans, dan jaringan game, termasuk bandwidth dan packet-per-second batas, menggunakan Amazon CloudWatch dan CloudWatch Synthetics untuk pemeriksaan fungsionalitas jalur kritis.
- GameLiftLacak metrik spesifik seperti GameServerInterruptionsdan ServerProcessAbnormalTerminationsuntuk menilai dampak ketersediaan instans Spot dan mendeteksi penghentian server yang tidak normal.

- Ekspor CloudWatch metrik ke Amazon S3 untuk penyimpanan jangka panjang, gunakan tingkatan hemat biaya seperti S3 Intelligent-Tiering atau Glacier, dan analisis tren dengan alat seperti Amazon Athena.

Sumber daya

- Metrik kinerja jaringan EC2 tingkat instans Amazon mengungkap wawasan baru
- CloudWatchAliran Metrik — Kirim AWS Metrik ke Mitra dan Aplikasi Anda secara Real Time

## Sumber daya

Lihat sumber daya berikut untuk mempelajari lebih lanjut tentang praktik terbaik kami terkait keandalan.

Dokumen terkait:

- Mempraktikkan Integrasi Berkelanjutan dan Pengiriman Berkelanjutan AWS
- Antrian Pekerjaan Asinkron Penskalaan Otomatis
- Desain WorkloadService Arsitektur Anda
- Batas waktu, percobaan ulang, dan backoff dengan jitter
- Kerangka Well-Architected - Pilar Keandalan
- Arsitektur untuk Skalabilitas yang Andal
- Perpustakaan Pembangun Amazon
- Pesan Real-Time Skala Besar untuk Game Multiplayer
- Pengantar Pola Pengembangan Game yang Dapat Diskalakan di AWS
- Menjalankan Layanan Mikro Kontainer di AWS
- Aplikasi Web Hostdi Cloud
- Membangun Infrastruktur Jaringan Multi-VPC yang Dapat Diskalakan dan Aman

Video terkait:

- re:Invent 2020: Ubisoft: Membangun game multipemain multi-platform AWS
- re:Invent 2018: Game Seluler Penskalaan Supercell

- re:invent 2019: Bagaimana CAPCOM membangun game menyenangkan dengan wadah, data, dan ML
- re:Invent 2018: Mengglobalisasi Akun Pemain di Riot Games Sambil Mempertahankan Ketersediaan
- re:invent 2020: GameLoft - Penyelaman mendalam migrasi danau data downtime nol

Pelatihan terkait:

- Menggunakan Amazon GameLiftFleet IQ untuk Server Game
- Hosting Server Game dengan Amazon EC2

# Efisiensi kinerja

Pilar efisiensi kinerja berfokus pada penggunaan sumber daya komputasi yang efisien untuk memenuhi persyaratan dan mempertahankan efisiensi itu seiring perubahan permintaan dan teknologi berkembang.

Ambil pendekatan berbasis data untuk memilih arsitektur berkinerja tinggi. Kumpulkan data komprehensif tentang arsitektur, dari desain tingkat tinggi hingga pemilihan dan konfigurasi jenis sumber daya. Pertimbangkan pilihan arsitektur Anda secara siklus untuk memanfaatkan serangkaian layanan dan solusi yang terus berkembang. Metrik membantu memahami penyimpangan dari kinerja yang diharapkan sehingga Anda dapat mengambil tindakan. Pendekatan berbasis data membantu melakukan pengorbanan dengan arsitektur Anda untuk meningkatkan kinerja, mengurangi biaya, atau meningkatkan pengalaman pengembang.

Area fokus

- Prinsip desain
- Pemilihan arsitektur
- Pemilihan wilayah
- Pengembangan berulang
- Komputasi dan perangkat keras
- Pemilihan komputasi
- Manajemen data
- Jaringan dan Pengiriman Konten
- Proses dan budaya
- Sumber daya

## Prinsip desain

Selain prinsip-prinsip desain dalam whitepaper AWS Well-Architected Framework, prinsip-prinsip desain berikut dapat mencapai efisiensi kinerja untuk game Anda:

- Ukur kinerja permainan dari end-to-end: Penting untuk mengukur kinerja seperti yang dirasakan dari perspektif pemain Anda. Ini berarti Anda harus mengukur kinerja klien game, infrastruktur game Anda, dan konektivitas internet yang menghubungkan pemain Anda ke infrastruktur. Ini akan

membantu dalam memahami di mana Anda dapat melakukan peningkatan kinerja dalam arsitektur Anda.

- Optimalkan arsitektur Anda untuk meningkatkan metrik yang mencerminkan pengalaman pemain yang sebenarnya: Saat Anda beradaptasi dan mengembangkan arsitektur Anda dari waktu ke waktu, pikirkan bagaimana peningkatan dan perubahan ini akan memengaruhi pengalaman pemain. Beban kerja game harus mampu menahan dan meminimalkan dampak kegagalan untuk memblokir gangguan gameplay yang meluas. Fitur dan sistem permainan yang tidak terlalu bergantung satu sama lain harus di-de-coupled untuk mengurangi radius ledakan kegagalan dan mengisolasi masalah yang berdampak pada pemain.
- Gunakan teknologi yang menyederhanakan operasi game dan meningkatkan kecepatan pengembangan: Prioritaskan adopsi teknologi yang dapat meningkatkan efisiensi pengembang. Overhead operasional selama tahap pengembangan pra-produksi dapat menjadi gangguan dari peningkatan gameplay. Dengan memanfaatkan layanan terkelola dari AWS atau AWS Mitra, teknik dapat dikurangi, yang memungkinkan pengembang game untuk fokus pada loop game inti dan pengalaman pemain. Persyaratan arsitektur dan kinerja dapat berubah dan berkembang sepanjang siklus hidup pengembangan game dan pertukaran teknologi harus dipertimbangkan pada setiap fase.
- Merancang infrastruktur untuk memenuhi konkurensi pemain puncak dan skala dinamis sesuai kebutuhan: Infrastruktur harus dirancang untuk skala untuk memenuhi permintaan pemain. Metrik, seperti konkurensi sesi pemain dan jumlah login, dapat digunakan untuk skala terlebih dahulu sebelum sistem menjadi kelebihan beban. Metrik pemanfaatan sistem reaktif, seperti CPU dan konsumsi memori, dapat digunakan untuk skala setelah sistem menjadi kelebihan beban. Dengan menskalakan infrastruktur Anda secara dinamis, Anda dapat mengurangi biaya pengoperasian game Anda.

# Pemilihan arsitektur

GAMEPERF01: Bagaimana Anda memilih opsi hosting yang sesuai untuk server game Anda?

Pemilihan opsi hosting yang sesuai untuk server game Anda adalah dasar untuk kinerja server game. Memutuskan untuk menggunakan EC2 instance, solusi kontainer, atau layanan yang dikelola sepenuhnya adalah salah satu keputusan pertama yang harus dibuat saat merancang untuk produksi. Setiap opsi hosting akan memiliki berbagai kemampuan dan pertimbangan untuk penyetelan kinerja, penskalaan, operasi, dan integrasi.

Praktik terbaik

- GAMEPERF01-BP01 Mengevaluasi persyaratan sumber daya server game dan kebutuhan skalabilitas
- GAMEPERF01-BP02 Pertimbangkan overhead operasional untuk penskalaan server game
- GAMEPERF01-BP03 Mengevaluasi integrasi dengan AWS layanan lain, lingkungan pengembangan, arsitektur CPU target, dan fitur

## GAMEPERF01-BP01 Mengevaluasi persyaratan sumber daya server game dan kebutuhan skalabilitas

Evaluasi persyaratan server terhadap kebutuhan skalabilitas Anda untuk memverifikasi bahwa Anda memilih opsi hosting yang memenuhi persyaratan Anda dan memberikan kinerja yang optimal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Saat memilih opsi hosting yang sesuai untuk server game Anda, pertimbangkan faktor-faktor berikut:

Persyaratan sumber daya server game

Nilai CPU, memori, jaringan, dan persyaratan penyimpanan dari proses server game Anda untuk menentukan apa yang dikonsumsi game Anda. Jangan mengabaikan jaringan; setiap frame membutuhkan siklus CPU untuk menerima tindakan pemain, memperbarui status permainan, dan mengirimkannya kembali ke pemain. Pemrosesan paket pembongkaran dapat membebaskan CPU untuk fungsi permainan inti. Jaringan adalah fondasi untuk permainan game yang lancar dan responsif sehingga mengujinya di awal proses menentukan profil kinerja dasar untuk sebuah game.

Game penembak orang pertama mungkin memiliki aksi tinggi per detik yang dibutuhkan CPU untuk segera beralih ke jaringan yang mungkin mendukung instance keluarga C yang dioptimalkan komputasi, sementara game strategi berbasis giliran yang dapat menghabiskan lebih banyak pemrosesan siklus CPU per giliran mungkin memerlukan peningkatan memori dari instance keluarga R untuk menyimpan dan memperbarui status game secara lokal di server sebelum mengirimnya kembali ke pemain. Gunakan pendekatan berbasis data seperti Metode Utilization Saturation and Errors (USE) untuk membuat pilihan arsitektur yang terinformasi dengan baik.

Skalabilitas dan elastisitas

Evaluasi seberapa cepat dan lancar setiap opsi hosting dapat disesuaikan untuk memenuhi permintaan pemain tanpa mengorbankan kinerja. Pertimbangkan tingkat otomatisasi dan fleksibilitas yang diperlukan untuk beban kerja game Anda untuk mempertahankan pengalaman bermain game yang lancar selama waktu puncak. Server game dapat menskalakan dengan cepat dengan meningkatkan pemanfaatan melalui penambahan proses server game tambahan pada contoh yang sama, di mana backend game dapat menskalakan lebih lambat berdasarkan peningkatan jumlah pengguna aktif dan game yang dimainkan. Armada Anda harus menskalakan dengan permintaan untuk meminimalkan biaya sambil memfasilitasi waktu tunggu minimal bagi para pemain untuk masuk ke permainan. Tinjau Amazon EC2 Spot Instance Advisor untuk mendapatkan wawasan tentang kapasitas hemat biaya yang tersedia untuk armada server game.

Langkah-langkah implementasi

- Evaluasi persyaratan sumber daya server game untuk CPU, memori, jaringan, dan penyimpanan untuk memilih jenis instans yang sesuai, dengan mempertimbangkan kebutuhan kinerja khusus game seperti throughput jaringan tinggi untuk game FPS atau pengoptimalan memori untuk game strategi berbasis giliran.
- Bandingkan opsi hosting yang berbeda seperti kontainer, instance, bare-metal, dan layanan terkelola dengan menganalisis data kinerja menggunakan kerangka kerja seperti metode USE. Gunakan wawasan ini untuk membuat keputusan yang lebih baik tentang arsitektur sistem Anda.
- Rancang armada untuk skalabilitas dan elastisitas, memanfaatkan alat seperti EC2 Spot Instance Advisor untuk mengoptimalkan biaya sekaligus memfasilitasi penskalaan cepat untuk memenuhi permintaan pemain selama waktu puncak.

## GAMEPERF01-BP02 Pertimbangkan overhead operasional untuk penskalaan server game

Pertimbangkan manajemen dan overhead operasional yang terkait dengan setiap opsi hosting.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Overhead operasional

Solusi yang di-host sendiri pada EC2 atau kontainer dapat memberikan kontrol lebih tetapi juga akan membutuhkan lebih banyak manajemen. Orkestrator kontainer seperti ECS atau EKS dapat

mengurangi waktu peluncuran untuk server kontainer sementara juga meningkatkan kompleksitas jaringan dan overhead orkestrasi pemeliharaan.

Sebagai contoh, grup node terkelola EKS dapat mengotomatiskan penyediaan dan manajemen siklus hidup server game Anda tetapi tidak menghormati anggaran gangguan pod saat menghentikan node, jika game Anda membutuhkan lebih lama dari periode terminasi 15 menit untuk menyelesaikan game dengan aman, Anda mungkin perlu membuat kait siklus hidup atau mempertimbangkan node yang dikelola sendiri dengan pengontrol khusus untuk memblokir gangguan game.

Layanan terkelola seperti Amazon Game Lift dapat menangani sebagian besar overhead operasional tetapi mengurangi jumlah visibilitas dan kontrol atas persyaratan khusus untuk jaringan tingkat rendah dan konfigurasi keamanan. Memilih solusi server game adalah trade-off antara tingkat penyesuaian, kontrol, dan tanggung jawab yang akan Anda miliki untuk menyetel kinerja server game dan perilaku penskalaan.

Langkah-langkah implementasi

- Nilai overhead operasional untuk opsi hosting, menyeimbangkan kontrol, dan upaya manajemen antara solusi yang dihosting sendiri seperti EC2, ECS, atau EKS dan layanan terkelola seperti Amazon Game Lift.
- Gunakan grup node terkelola EKS untuk otomatisasi tetapi terapkan kait siklus hidup atau pengontrol khusus jika server game Anda memerlukan periode penghentian yang lebih lama daripada default.
- Timbang trade-off antara kustomisasi, visibilitas, dan tanggung jawab operasional saat memilih solusi server game.

## GAMEPERF01-BP03 Mengevaluasi integrasi dengan AWS layanan lain, lingkungan pengembangan, arsitektur CPU target, dan fitur

Evaluasi seberapa baik setiap opsi hosting terintegrasi dengan AWS layanan lain yang diandalkan game Anda, seperti database, analitik, atau layanan pengiriman konten.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Integrasi dengan AWS layanan lain

Integrasi yang mulus antar layanan memberikan manfaat operasional seperti pemantauan kinerja yang ditingkatkan dan pengiriman data aman yang efisien antara komponen game, server game, layanan backend game, dan solusi observabilitas.

Misalnya, Mengkoordinasikan pergeseran lalu lintas untuk permainan langsung bisa jadi rumit. Amazon Route 53 akan membantu menjaga catatan DNS Anda tetap up to date yang menyederhanakan pergeseran lalu lintas terkoordinasi. AWS Dial lalu lintas Global Accelerator memungkinkan Anda mengirim persentase lalu lintas ke Wilayah lain dan menjaga permainan Anda tetap berjalan selama pemeliharaan.

Lingkungan dan alat pengembangan

Pertimbangkan alat pengembangan, kerangka kerja, dan lingkungan yang didukung oleh setiap opsi arsitektur. Verifikasi bahwa opsi yang Anda pilih selaras dengan solusi pengembangan game dan bahasa pemrograman Anda, karena ini dapat memengaruhi kemampuan tim Anda untuk mengoptimalkan dan mempertahankan kinerja server game. Menghadirkan game di ponsel, konsol, dan PC akan meningkatkan kompleksitas perkakas dan pengujian. Dukungan lintas sistem sangat penting untuk studio multi-game di mana layanan terpusat dapat menstandarisasi praktik terbaik pengembangan di seluruh judul.

Arsitektur dan fitur CPU target

Pertimbangkan profil kinerja mesin game dan proses server game Anda dan tingkat dukungan ARM yang tersedia. Evaluasi apakah Anda dapat memperoleh manfaat dari peningkatan kinerja harga prosesor berbasis Graviton atau x86 berbasis ARM. AMD64 Apakah Anda perlu menggunakan fitur Intel seperti enkripsi AES-NI, AVX atau Turbo Boost? Tinjau jenis Host Khusus untuk mengidentifikasi keluarga instans tunggal versus multi-soket. Saat menggunakan keluarga instans multi-soket, pertimbangkan penyematan NUMA dan berbagi cache L3 dalam proses server game Anda. Gunakan konfigurasi C-state dan P-state untuk mendapatkan kinerja terbaik untuk game Anda dengan menyetel jam frekuensi dan mengurangi tingkat tidur.

Langkah-langkah implementasi

- Pilih opsi hosting dengan integrasi tanpa batas dengan AWS layanan seperti AWS Secrets Manager, ACM, dan lainnya untuk membantu merampingkan pemantauan kinerja, mengamankan pengiriman data, dan mengurangi tugas operasional manual.
- Verifikasi kompatibilitas antara opsi hosting Anda dan lingkungan pengembangan, kerangka kerja, dan bahasa pemrograman Anda untuk mengoptimalkan dan mempertahankan kinerja server secara efektif.

- Evaluasi persyaratan arsitektur CPU, manfaatkan Graviton untuk kinerja harga atau x86 untuk fitur spesifik seperti AES-NI, AVX, dan Turbo Boost, dan optimalkan kinerja server dengan pinning NUMA dan penyetelan C-state/P-state.

# Pemilihan wilayah

GAMEPERF02: Bagaimana Anda menentukan Wilayah geografis mana yang menjadi tuan rumah infrastruktur game Anda?

Memilih lokasi yang ideal untuk infrastruktur game Anda dapat meningkatkan kinerja jaringan untuk pemain dan backend. Mempertimbangkan dari mana basis pemain Anda terhubung dan bagaimana komunitas atau server Anda dibangun penting untuk pertumbuhan jangka panjang dan keberlanjutan di Wilayah geografis. Menyebarkan infrastruktur server game dan layanan backend yang dipisahkan dapat menguntungkan efisiensi operasional Anda secara keseluruhan dan meningkatkan ketahanan dengan menggunakan beberapa Wilayah, zona lokal, dan pos terdepan untuk meng-host game Anda.

Praktik terbaik

- GAMEPERF02-BP01 Pilih Wilayah rumah yang dekat dengan pemain Anda
- GAMEPERF02-BP02 Merancang pendekatan yang mendukung penempatan infrastruktur game yang sensitif terhadap latensi dekat dengan pemain untuk meningkatkan kinerja

## GAMEPERF02-BP01 Pilih Wilayah rumah yang dekat dengan pemain Anda

Untuk peluncuran game awal, Anda harus menentukan di mana harus menyebarkan infrastruktur berdasarkan diskusi dengan pemangku kepentingan bisnis Anda, seperti tim penerbitan yang menentukan di mana game diharapkan tersedia untuk pemain, dan di mana mereka memfokuskan pemasaran pra-peluncuran dan upaya periklanan mereka.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Pemangku kepentingan bisnis Anda juga harus memiliki mekanisme untuk merangsang permintaan untuk mendapatkan pemahaman yang lebih baik tentang penerimaan dan kelayakan pemain.

Misalnya, tim-tim ini akan memiliki mekanisme seperti pre-order game, acara pemasaran dan kampanye, daftar email publik bagi pemain untuk mendaftarkan minat sebelum diluncurkan, dan pendekatan lain untuk menetapkan sinyal yang relevan untuk menentukan di mana permainan kemungkinan akan memiliki pemain terbanyak saat diluncurkan. Permainan ini juga dapat menggunakan strategi peluncuran regional yang mencakup fase uji permainan dan peluncuran lunak untuk menentukan permintaan pemain regional.

Pilih Wilayah rumah yang dekat dengan basis pemain Anda dan pengembang Anda dan memiliki AWS layanan dan fitur yang Anda butuhkan untuk meng-host game Anda. Rumah RSegion akan menjadi tempat layanan backend game akan berjalan, dan juga dapat menjalankan server game. Evaluasi Wilayah asal berdasarkan layanan yang didukung, konektivitas ke lokasi tepi, kedekatan dengan Wilayah failover, dan jumlah Availability Zone. Jika Anda menggunakan Zona Lokal, pertimbangkan Wilayah induk terkadang terletak di wilayah geografis yang berbeda. Sebagai contoh: Santiago, Zona Lokal Chili us-timur-1-scl-1a memiliki N. Virginia us-east-1 sebagai Wilayah induknya meskipun secara geografis lebih dekat ke Sao Paulo sa-east-1.

Langkah-langkah implementasi

- Identifikasi Wilayah penyebaran berdasarkan sinyal permintaan pemain dari aktivitas pra-peluncuran seperti pre-order, kampanye pemasaran, dan pendaftaran minat.
- Pilih Wilayah rumah yang dekat dengan basis pemain utama dan pengembang, pastikan itu mendukung AWS layanan yang diperlukan, lokasi tepi, dan Wilayah failover.
- Evaluasi Local Zones dengan hati-hati, mengingat bahwa Wilayah induk mungkin berbeda secara geografis dari lokasi Zona Lokal.

## GAMEPERF02-BP02 Merancang pendekatan yang mendukung penempatan infrastruktur game yang sensitif terhadap latensi dekat dengan pemain untuk meningkatkan kinerja

Penempatan terpisah untuk infrastruktur sensitif latensi seperti server game meminimalkan dampak rute jaringan yang panjang. Penerapan berulang dapat membuatnya mudah untuk mempertahankan beberapa lokasi yang lebih berkinerja untuk pemain Anda. Ping adalah metrik umum yang muncul di UI game dan ping rendah dapat menjadi kemampuan pembeda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Saat pertama kali meluncurkan game, Anda mungkin belum memiliki informasi yang cukup tentang basis pemain Anda untuk mengetahui secara memadai di mana terbaik untuk menyebarkan infrastruktur yang paling dekat dengan pemain yang paling tertarik untuk memainkan game Anda. Ini adalah tantangan umum, dan Anda harus mempersiapkan skenario ini dengan merancang arsitektur yang memungkinkan Anda untuk dengan cepat menyesuaikan strategi penempatan hosting Anda untuk menyebarkan server di mana mereka dibutuhkan lebih dekat dengan pemain. Adalah tipikal bagi pengembang game untuk secara teratur menilai penyebaran infrastruktur game mereka sebagai analisis pasca-peluncuran berulang untuk berinvestasi secara bertahap dalam peningkatan dari waktu ke waktu dengan pendekatan berulang.

Praktik terbaik adalah menggunakan infrastructure-as-code templat, seperti AWS CloudFormation atau Terraform by Hashicorp, untuk konfigurasi infrastruktur Anda seperti, konfigurasi subnet VPCs, dan dependensi yang diperlukan untuk meluncurkan layanan game penting sehingga Anda dapat merujuk ke templat ini, dengan cepat menyesuaikannya jika diperlukan, dan menerapkannya ke lokasi di mana infrastruktur tambahan diperlukan untuk mendukung pemain Anda.

Anda juga harus memastikan bahwa Anda memahami bagaimana strategi penyebaran Anda saat ini dapat dikembangkan untuk memungkinkan ekspansi di masa depan. Template IAC dapat diulang tetapi bukan pengganti perencanaan jaringan. IPAM mengelola Anda VPCs. Ukuran subnet, pemilihan Availability Zone, dan inventaris IP dan penyelarasan Availability Zone lintas akun. Jaringan penting untuk dipertimbangkan dan dapat mengganggu pemain saat diubah. Server game yang digunakan di beberapa lokasi geografis akan terhubung ke backend game Anda, yang lebih umum dihosting di satu atau beberapa Wilayah rumah yang memerlukan konfigurasi tambahan untuk mendukung konektivitas pribadi. Pertimbangan ini harus terus dievaluasi dari waktu ke waktu sehingga Anda dapat membuat perubahan pada strategi hosting game Anda saat persyaratan game Anda berkembang atau persyaratan pemain Anda berubah.

Saat menentukan berapa banyak lokasi hosting game yang akan digunakan untuk game Anda, pertimbangkan faktor-faktor berikut:

- Peningkatan kualitas pengalaman pemain: Berapa banyak peningkatan pengalaman pemain yang dapat Anda perkenalkan dengan menambahkan lokasi hosting game tambahan? Apa keuntungan kinerja tambahan yang dapat Anda capai dengan melakukannya? Bagaimana Anda mengukur peningkatan kinerja ini?

- Populasi pemain mana yang harus diprioritaskan: Berapa banyak pemain yang dapat Anda tingkatkan pengalaman jika Anda menambahkan lokasi hosting game tambahan? Populasi pemain mana, atau lokasi geografis, yang akan Anda prioritaskan?
- Dampak perubahan hilir: Jika Anda mengubah strategi hosting game Anda, bagaimana ini akan memengaruhi waktu tunggu perjodohan Anda untuk pemain? Bisakah ukuran pertandingan, keseimbangan keterampilan, atau jumlah pemain di kolam pemain mengakomodasi perubahan strategi lokasi hosting game? Mendukung lebih banyak lokasi berpotensi memecah kumpulan pemain dan menambah peningkatan biaya dan kompleksitas.

Masing-masing pertimbangan ini harus dievaluasi saat Anda menentukan di mana Anda menambahkan atau menghapus lokasi hosting game. Misalnya, Anda dapat memilih untuk memprioritaskan peningkatan pengalaman bagi pemain di lokasi geografis dengan pengalaman gameplay yang paling tidak berkinerja, atau untuk pemain yang mengekspresikan umpan balik publik paling vokal. Anda juga dapat memilih untuk memasukkan monetisasi pemain ke dalam prioritas Anda, misalnya dengan memusatkan perhatian pada peningkatan pengalaman bagi pemain di lokasi geografis yang menghasilkan sumber pendapatan yang signifikan untuk permainan Anda atau memiliki potensi untuk menghasilkan pendapatan tambahan jika Anda memperkenalkan peningkatan kinerja.

Selain menghosting infrastruktur di Wilayah AWS, Anda dapat menggunakan Local Zones, yang merupakan perpanjangan dari Wilayah AWS, untuk meng-host server game Anda dan aplikasi sensitif latensi lainnya seperti server obrolan suara yang lebih dekat dengan pemain Anda. Anda juga dapat memilih untuk menjalankan infrastruktur pengembangan game di Local Zones untuk meningkatkan pengalaman bagi tim pengembangan game Anda. Misalnya, Anda dapat menggunakan Local Zones untuk mengatasi kasus penggunaan seperti menghosting replika server kontrol sumber yang dikelola sendiri lebih dekat dengan pengembang game, dan untuk menawarkan workstation virtual pengembangan game dan penyimpanan konten kepada pengguna yang menggunakan EC2 instans Amazon, volume EBS, dan sistem FSx file Amazon yang diterapkan ke satu atau beberapa Local Zones di dekat studio pengembangan Anda tanpa mengharuskan Anda meng-host infrastruktur lokal.

Outposts adalah pilihan yang baik ketika Wilayah atau Local Zones tidak tersedia di wilayah geografis yang sama. Konektivitas dari pusat data Anda AWS harus dipertimbangkan untuk memungkinkan server game untuk keandalan sistem backend. AWS Outposts dan Outpost Server dibuat khusus untuk dijalankan AWS di pusat data Anda menggunakan layanan yang sama dan APIs untuk membantu membuat model penerapan yang konsisten di mana pun Anda menjalankan game.

Beberapa rak dapat digabungkan menjadi Outpost logis, dan infrastruktur dapat dibagi. Akun AWS Siklus hidup perangkat keras dikelola oleh AWS dan waktu tunggu bisa sesingkat 3 bulan.

Jika Anda membangun game menggunakan kontainer dan menginginkan fleksibilitas untuk mengadopsi arsitektur penyebaran hibrida menggunakan perangkat lunak sumber terbuka yang dapat digunakan di infrastruktur lokal Anda sendiri, Anda dapat menggunakan ECS Anywhere, atau EKS Anywhere sebagai alternatif atau Local Zones. AWS Outposts Jika Anda meng-host dengan Amazon GameLift; Amazon GameLift Anywhere dapat digunakan untuk menjalankan server Anda di perangkat keras lokal yang dapat mempercepat proses pengembangan Anda, memungkinkan Anda untuk menggunakan Local Zones atau mendaftarkan logam Anda sendiri sebagai bagian dari armada Anda.

Langkah-langkah implementasi

- Gunakan infrastructure-as-code alat seperti AWS CloudFormation atau Terraform untuk penerapan berulang, memungkinkan penyesuaian cepat dan penskalaan lokasi hosting game berdasarkan kebutuhan pemain.
- Evaluasi peningkatan pengalaman pemain, prioritas populasi pemain, dan dampak hilir seperti waktu perjodohan saat menambahkan atau menghapus lokasi hosting game.
- Gunakan AWS Local Zones, Outposts, atau opsi hybrid seperti ECS Anywhere, EKS Anywhere, atau GameLift Anywhere untuk mengoptimalkan infrastruktur yang sensitif terhadap latensi dan mendukung beragam kebutuhan penerapan.

## Pengembangan berulang

GAMEPERF03: Bagaimana Anda bisa menggunakan Amazon GameLift untuk efisiensi kinerja pengembangan berulang?

Amazon GameLift menyediakan end-to-end alur kerja untuk pengembangan dan pengujian kinerja game Anda di lingkungan pengujian lokal.

Praktik terbaik

- GAMEPERF03-BP01 Gunakan Amazon GameLift Anywhere dan toolkit pengujian GameLift
- GAMEPERF03-BP02 Uji kinerja dan skalabilitas server game
- GAMEPERF03-BP03 Optimalkan pemanfaatan sumber daya kontainer GameLift

# GAMEPERF03-BP01 Gunakan Amazon GameLift Anywhere dan toolkit pengujian GameLift

Untuk meningkatkan efisiensi kinerja melalui proses pengembangan berulang, gunakan Amazon GameLift Anywhere bersama dengan Amazon GameLift Testing Toolkit untuk membangun lingkungan pengujian yang komprehensif.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Pendekatan ini memungkinkan iterasi cepat, pengumpulan data yang efisien, dan analisis kinerja terperinci. Langkah-langkah kunci meliputi:

Buat lingkungan pengujian

Gunakan Amazon GameLift Anywhere untuk menyiapkan lingkungan pengujian lokal atau berbasis cloud. Pengaturan ini menghilangkan kebutuhan untuk mengunggah setiap iterasi build server game ke armada terkelola, sehingga mengurangi waktu aktivasi.

Integrasikan Toolkit GameLift Pengujian Amazon

Menggabungkan Amazon GameLift Testing Toolkit ke dalam alur kerja pengembangan Anda. Toolkit ini menyediakan skrip, alat, dan pustaka untuk memvisualisasikan GameLift infrastruktur Amazon, meluncurkan pemain virtual, dan mengulangi set FlexMatch aturan dengan simulator. FlexMatch Ini menyederhanakan integrasi dan pengelolaan GameLift sumber daya Amazon, memungkinkan Anda untuk mengotomatiskan tugas-tugas umum dan mengumpulkan data yang diperlukan untuk analisis kinerja.

Siklus pembuatan dan pengujian yang cepat

Perbarui armada uji dengan cepat dengan build baru, mulai, dan mulai pengujian. Ini memfasilitasi build-test-repeat siklus cepat, memungkinkan pengembang untuk memvalidasi berbagai aspek pengalaman pemain game, termasuk interaksi multipemain.

Pengujian komprehensif

Uji integrasi server game Anda dengan SDK GameLift server Amazon, interaksi layanan backend, konfigurasi perjodohan, dan fitur hosting lainnya. GameLift Manfaatkan GameLift Testing Toolkit untuk mengotomatiskan pengujian dan mengumpulkan metrik kinerja terperinci, memastikan bahwa komponen game bekerja sama dengan mulus.

Menganalisis data kinerja

Gunakan data yang dikumpulkan oleh GameLift Testing Toolkit untuk menganalisis kemacetan kinerja dan mengoptimalkan server game Anda. Toolkit membantu melacak metrik utama, mengidentifikasi masalah, dan membuat keputusan berbasis data untuk meningkatkan efisiensi kinerja.

Dengan menggabungkan Amazon GameLift Anywhere dan GameLift Testing Toolkit ke dalam proses pengembangan berulang, Anda dapat secara signifikan meningkatkan efisiensi kinerja melalui pengujian cepat, pemeriksaan integrasi komprehensif, dan analisis kinerja terperinci.

Langkah-langkah implementasi

- Gunakan Amazon GameLift Anywhere untuk membuat lingkungan pengujian, mengurangi waktu aktivasi untuk pembuatan server game, dan mengaktifkan iterasi cepat.
- Integrasikan Amazon GameLift Testing Toolkit untuk mengotomatiskan tugas pengujian, mensimulasikan pemain, dan memvalidasi konfigurasi selama pengembangan FlexMatch.
- Kumpulkan dan analisis data kinerja dengan GameLift Testing Toolkit untuk mengidentifikasi kemacetan, mengoptimalkan server game, dan meningkatkan efisiensi kinerja.

## GAMEPERF03-BP02 Uji kinerja dan skalabilitas server game

Untuk menguji kinerja dan skalabilitas server game Anda, terapkan kerangka pengujian yang kuat menggunakan GameLift fitur Amazon dan Toolkit GameLift Pengujian.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Praktik utama meliputi:

Pengujian berulang

Gunakan armada Amazon GameLift Anywhere untuk membuat lingkungan host berbasis cloud tempat Anda dapat membuat dan menguji komponen game secara berulang. Lingkungan ini harus mencerminkan kondisi hosting dunia nyata, memungkinkan kinerja realistis dan pengujian skalabilitas.

Pengujian integrasi server game

Uji integrasi server game Anda dengan SDK GameLift server Amazon, termasuk memulai sesi permainan baru dan melacak peristiwa sesi game menggunakan AWS CLI atau GameLift Testing Toolkit. Ini memverifikasi bahwa server game berfungsi dengan benar di dalam GameLift lingkungan.

Gunakan GameLift Testing Toolkit untuk mengotomatiskan pengujian dan mengumpulkan metrik kinerja terperinci. Toolkit ini memungkinkan Anda untuk memvisualisasikan GameLift infrastruktur, meluncurkan pemain virtual untuk pengujian beban, dan mengulangi set FlexMatch aturan dengan simulator. FlexMatch Ini sangat berguna untuk penskalaan tugas ECS Fargate, yang mensimulasikan sesi pemain dengan membuat banyak sesi permainan bersamaan untuk menguji stres infrastruktur server.

Pengujian skalabilitas

Bereksperimenlah dengan desain antrian sesi game, armada multi-lokasi, armada Spot dan On-Demand, dan beberapa jenis instans. Uji opsi penempatan sesi game, kebijakan latensi, dan pengaturan prioritas armada. Konfigurasikan penskalaan kapasitas untuk memenuhi permintaan pemain dan validasi bahwa sistem dapat menangani beban yang diharapkan dalam kondisi yang berbeda.

Langkah-langkah implementasi

- Gunakan Amazon GameLift Anywhere untuk menyiapkan lingkungan pengujian realistis untuk pengujian kinerja dan skalabilitas berulang.
- Uji integrasi server game dengan SDK GameLift server, memfasilitasi manajemen sesi yang benar dan pelacakan acara dalam lingkungan. GameLift
- Lakukan pengujian skalabilitas dengan GameLift Testing Toolkit, simulasi pemuatan pemain, uji antrian sesi, dan validasi penskalaan armada, kebijakan latensi, dan pengaturan prioritas.

## GAMEPERF03-BP03 Optimalkan pemanfaatan sumber daya kontainer GameLift

Untuk mengoptimalkan pemanfaatan sumber daya GameLift kontainer, rancang armada kontainer Anda secara efektif dan tetapkan batas sumber daya yang tepat.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Pedoman utama meliputi:

- Desain grup kontainer: Atur perangkat lunak Anda ke dalam grup kontainer. Wadah utama harus menggabungkan aplikasi server game Anda dan GameLift Agen Amazon. Gunakan wadah sidecar untuk perangkat lunak tambahan untuk mengelola dependensi dan menetapkan batas khusus container untuk penggunaan memori dan CPU.
- Tetapkan batas sumber daya: Untuk setiap grup kontainer, tentukan memori dan sumber daya CPU yang diperlukan. Tetapkan batas opsional untuk kontainer individu untuk memverifikasi bahwa mereka memiliki sumber daya cadangan tetapi juga dapat melebihi batas ini jika sumber daya tambahan tersedia. Ini membantu mencegah perselisihan sumber daya dan potensi kegagalan kontainer.
- Grup kontainer daemon: Pertimbangkan untuk menggunakan grup kontainer daemon untuk latar belakang atau proses pemantauan yang tidak perlu diskalakan dengan grup kontainer utama. Ini memverifikasi bahwa tugas latar belakang penting ditangani secara efisien tanpa memengaruhi proses server game utama.

Langkah-langkah implementasi

- Rancang grup kontainer dengan wadah utama untuk server game dan GameLift Agen, dan sidecars untuk mengelola dependensi, dengan memori dan batas CPU tertentu.
- Tetapkan batas sumber daya untuk setiap grup kontainer untuk menyimpan sumber daya yang diperlukan sambil memungkinkan penggunaan sumber daya terkontrol untuk menghindari pertengkaran.
- Gunakan grup kontainer daemon untuk tugas latar belakang atau pemantauan, pastikan mereka beroperasi secara efisien tanpa mempengaruhi proses server game utama.

## Komputasi dan perangkat keras

GAMEPERF04: Bagaimana Anda memblokir sesi game agar tidak memengaruhi pemain yang berjalan di instance server game yang sama?

Setelah server game Anda berjalan AWS, Anda perlu memantau kinerjanya untuk memberikan pengalaman pemain yang berkualitas terlepas dari pemanfaatan sumber daya, komputasi yang mendasarinya, atau saturasi.

Praktik terbaik

- GAMEPERF04-BP01 Memantau proses server game untuk mendeteksi masalah
- GAMEPERF04-BP02 Uji kinerja server game Anda dengan skenario gameplay simulasi dan nyata

## GAMEPERF04-BP01 Memantau proses server game untuk mendeteksi masalah

Anda dapat menjalankan beberapa proses server game per instance untuk memanfaatkan sumber daya secara efisien pada instance server game Anda. Jika demikian, rancang arsitektur Anda sehingga proses server game individual yang menghosting sesi game tidak dapat menyebabkan dampak buruk pada sesi game lain yang dihosting pada instance yang sama. Gunakan metrik untuk memahami bagaimana penempatan game dan jenis mode game dapat memengaruhi kinerja instance server game. Gabungkan campuran proses beban rendah (lobi, toko, atau tutorial pemain tunggal) dan beban tinggi (peringkat, multi-pemain, atau gameplay keterampilan tinggi) untuk menghindari hot spotting pada instance server game.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Pantau pengalaman pemain melalui metrik sisi klien dan sisi server dengan mengumpulkan telemetri untuk waktu ping dan jitter, penurunan bingkai, waktu respons API, kesalahan, dan penyelesaian loop game yang berhasil. Korelasikan stempel waktu untuk acara ini dengan masalah dukungan pemain dan log server untuk mengidentifikasi kemacetan kinerja. Alat seperti Dtrace, ftrace, uperf, dan eBPF dapat digunakan untuk penyelidikan mendalam dan analisis kinerja sistem.

Terapkan pemantauan sumber daya terbatas yang tersedia untuk instance server game Anda sehingga Anda dapat menghasilkan peringatan ketika proses server game individual melanggar ambang batas anggaran sumber daya yang telah ditentukan sebelumnya. Ketika ambang batas dilanggar, Anda mungkin ingin mengonfigurasi perangkat lunak server game Anda untuk membuang log sistem dan server game yang relevan ke penyimpanan yang tahan lama, seperti solusi logging pusat, sehingga insinyur server game Anda dapat menyelidiki perilaku ini. Selain itu, instance server game Anda harus dikonfigurasi untuk melaporkan metrik dari setiap proses server game yang berjalan pada instance sehingga Anda dapat memantau proses server game individual ini selain metrik keseluruhan untuk instance server game.

Misalnya, GameLift menyediakan metrik untuk memantau sesi game, yang dapat ditambah dengan metrik khusus game khusus dan log yang dikumpulkan menggunakan CloudWatch Agen Amazon yang dapat Anda konfigurasikan pada instance server game Anda. Metrik Anda dapat dilihat

CloudWatch atau diekspor ke alat lain seperti Grafana Terkelola Amazon yang terintegrasi dengan Sistem Masuk Tunggal untuk memudahkan akses metrik oleh pengguna yang mungkin tidak memiliki akses ke Konsol Manajemen. Lihat praktik terbaik berikut untuk mengelola log dan metrik menggunakan Amazon GameLift, yang juga menyediakan dukungan untuk melihat log sesi game individual.

Langkah-langkah implementasi

- Jalankan beberapa proses server game per instance dan campur mode permainan dengan beban rendah dan tinggi untuk menghindari hot spotting dan memverifikasi pemanfaatan sumber daya yang seimbang.
- Pantau metrik sisi klien dan sisi server seperti ping, jitter, frame drop, dan waktu respons API, dan korelasikan ini dengan log server dan masalah yang dilaporkan oleh pemain untuk mengidentifikasi kemacetan.
- Konfigurasikan pemantauan sumber daya untuk setiap proses server game, buat peringatan untuk pelanggaran ambang batas, dan simpan log dalam penyimpanan tahan lama untuk analisis menggunakan alat seperti dan Grafana CloudWatch Terkelola Amazon.

## GAMEPERF04-BP02 Uji kinerja server game Anda dengan skenario gameplay simulasi dan nyata

Lakukan pengujian kinerja dan evaluasi berbagai skenario gameplay untuk menentukan apakah proses server game menangani pemanfaatan sumber daya tetap dengan tepat, seperti memori EC2 instance, CPU, dan bandwidth jaringan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Membuat tes gameplay simulasi dengan bot yang dapat mencerminkan jalur gameplay umum dan perilaku pemain Anda dapat menentukan bagaimana proses server game Anda menangani ini di bawah skenario penggunaan yang berbeda. Misalnya, Anda dapat menerapkan solusi, seperti Pengujian Beban Terdistribusi AWS yang dapat Anda sesuaikan untuk menjalankan simulasi klien game atau build klien game untuk menghasilkan skenario gameplay. Jalankan tes bermain internal dan gunakan tim QA untuk menguji stres berbagai fitur permainan Anda sehingga Anda dapat mengembangkan kepercayaan diri bahwa permainan Anda dirancang untuk tampil optimal. AWS Device Farmdapat digunakan untuk melakukan pengujian seluler dan web untuk game iOS, Android, dan browser Anda di beberapa jenis perangkat.

Langkah-langkah implementasi

- Lakukan pengujian kinerja dengan bot yang mensimulasikan perilaku pemain umum untuk mengevaluasi pemanfaatan sumber daya server game dalam skenario yang berbeda.
- Gunakan solusi seperti Distributed Load Testing on AWS untuk menyesuaikan dan mensimulasikan skenario gameplay untuk stress testing.
- Lakukan playtest internal dan gunakan alat seperti AWS Device Farm untuk pengujian game seluler dan browser di berbagai perangkat.

# Pemilihan komputasi

GAMEPERF05: Bagaimana Anda memilih solusi komputasi yang sesuai untuk game Anda?

Kinerja komputasi bervariasi di seluruh ukuran instans dan keluarga. Sangat bermanfaat untuk menggunakan beberapa opsi komputasi yang berasal dari kumpulan kapasitas terpisah. Kembangkan strategi komposisi armada yang memberikan preferensi pada kinerja tetapi mencakup keragaman yang cukup untuk menghindari kesalahan kapasitas yang tidak mencukupi.

Praktik terbaik

- GAMEPERF05-BP01 Benchmark performa game Anda di berbagai jenis komputasi
- GAMEPERF05-BP02 Pindahkan tugas komputasi ke alur kerja asinkron non-latency-sensitive

## GAMEPERF05-BP01 Benchmark performa game Anda di berbagai jenis komputasi

Untuk beban kerja server game, tidak ada pendekatan tunggal untuk mengidentifikasi solusi komputasi optimal untuk hosting server game Anda. Strategi umum untuk membandingkan server game adalah memulai dengan instance EC2 'c' yang dioptimalkan komputasi, karena rangkaian instance ini memberikan kinerja tinggi untuk beban kerja yang intensif secara komputasi. Atau, jika game Anda membutuhkan sejumlah besar memori untuk mengimplementasikan fitur tertentu, instance yang dioptimalkan memori mungkin paling cocok.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Jika beban kerja Anda menggunakan sumber daya jaringan yang signifikan, pertimbangkan untuk menerapkan instance yang dioptimalkan jaringan yang biasanya diindikasikan menggunakan 'n' dalam nama instance, hindari jenis instance burstable 't' karena setelah kredit habis kinerja akan menurun. Game sensitif terhadap latensi dan paket yang dijatuhkan, jadi disarankan untuk menggunakan jaringan yang EC2 disempurnakan untuk membantu meningkatkan kinerja jaringan server game Anda. Jaringan yang disempurnakan menggunakan I/O virtualisasi root tunggal (SR-IOV) untuk memberikan kemampuan jaringan berkinerja tinggi pada jenis instans yang didukung. SR-IOV adalah metode virtualisasi perangkat yang memberikan I/O kinerja yang lebih tinggi dan pemanfaatan CPU yang lebih rendah jika dibandingkan dengan antarmuka jaringan virtual tradisional. Jaringan yang ditingkatkan memberikan bandwidth yang lebih tinggi, performa paket per detik yang lebih tinggi (PPS), dan latensi antar-instans lebih rendah yang konsisten. Jaringan yang disempurnakan dengan Adaptor Jaringan Elastis tersedia untuk jenis EC2 instans terbaru dan penting untuk diperbarui secara berkala untuk mendapatkan manfaat dari peningkatan kinerja dari instans yang lebih baru dan peningkatan pada hypervisor Nitro.AWS

Jika game Anda berkinerja sama di beberapa jenis EC2 instans, maka Anda harus mempertimbangkan untuk menggunakan beberapa jenis instance untuk meng-host server game Anda. Pantau kinerja dari waktu ke waktu dan lakukan pengoptimalan lebih lanjut setelah Anda menyelenggarakan sesi permainan produksi yang cukup untuk dapat mengidentifikasi tren kinerja. Ingatlah bahwa persyaratan komputasi Anda dapat berubah saat Anda menambahkan fitur baru ke dalam game Anda yang memerlukan alokasi sumber daya yang berbeda. Anda dapat mengonfigurasi grup EC2 Auto Scaling untuk menggunakan beberapa jenis instans, atau Anda dapat menggunakan grup Auto Scaling terpisah untuk meng-host instance server game yang menjalankan jenis instans terpisah yang dapat mempermudah pengelolaan korelasi dan agregasi metrik.

Evaluasi kinerja game Anda pada berbagai jenis prosesor seperti instance berbasis Intel, instance berbasis AMD, dan instance Graviton berbasis ARM. Unreal Engine 5.1.1 atau yang lebih baru dapat mengkompilasi server game untuk Graviton dan dapat meningkatkan kinerja harga untuk game Anda. Lakukan pengujian sapuan dan saturasi pada berbagai ukuran dalam setiap keluarga untuk menentukan titik manis di mana pemanfaatan dan kinerja konsisten.

Anda juga harus membandingkan bagaimana kinerja game Anda terpengaruh saat di-host menggunakan wadah dan fungsi Lambda. Untuk kasus penggunaan di mana proses server game berumur panjang tidak diperlukan, seperti game asinkron dan untuk layanan backend game, pertimbangkan untuk menggunakan arsitektur tanpa server dengan Lambda yang dapat menyederhanakan manajemen dan operasi untuk tim operasi game, serta memungkinkan Anda

untuk lebih cepat menyebarkan game Anda secara global ke banyak orang. Wilayah AWS Untuk praktik terbaik tanpa server, lihat Lensa Aplikasi Tanpa Server - Kerangka Kerja Well-Architected.

Langkah-langkah implementasi

- Benchmark server game pada instans 'c' yang dioptimalkan komputasi untuk beban kerja intensif CPU, instance yang dioptimalkan memori untuk tugas berat memori, dan instans 'n' yang dioptimalkan jaringan untuk throughput jaringan yang tinggi.
- Gunakan jaringan yang disempurnakan dengan Elastic Network Adapter (ENA) pada instans yang didukung untuk meningkatkan kinerja jaringan, mengurangi latensi, dan meningkatkan laju pemrosesan paket.
- Evaluasi dan uji beberapa jenis instans, prosesor (Intel, AMD, Graviton), dan opsi hosting container atau Lambda, sesuaikan solusi komputasi saat fitur game berkembang.

Untuk informasi selengkapnya, lihat Memilih strategi komputasi yang tepat untuk server game global Anda.

## GAMEPERF05-BP02 Pindahkan tugas komputasi ke alur kerja asinkron non-latency-sensitive

Ketika Anda mengoptimalkan kinerja untuk permainan Anda, penting untuk diingat bahwa hanya beberapa interaksi antara klien dan backend game yang harus dilakukan secara sinkron. Anda harus mempertimbangkan setiap fitur dari perspektif pengalaman pemain dan menentukan apakah interaksi tertentu memerlukan komunikasi sinkron, yang memblokir dan sumber daya intensif, atau apakah fitur tersebut dapat diimplementasikan secara asinkron. Saat Anda menerapkan panggilan jaringan, gunakan pendekatan asinkron, non-pemblokiran. Selain itu, backend game Anda juga harus dikonfigurasi untuk melakukan pekerjaan secara efisien dengan membongkar tugas ke antrian dan memprioritaskan respons cepat kepada klien jika memungkinkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Misalnya, memperbarui papan pemimpin di akhir sesi pemain dapat diimplementasikan secara asinkron sehingga klien tidak perlu menunggu pembaruan papan pemimpin selesai. Sebagai gantinya, terapkan ini secara asinkron pada klien game, dan pertimbangkan untuk merancang layanan backend Anda untuk mendorong jenis operasi ini ke dalam antrian, seperti Amazon SQS. Dengan arsitektur ini, konfigurasikan backend Anda untuk menerima permintaan, masukkan ke SQS

yang membantu menyimpan pesan secara tahan lama untuk pemrosesan asinkron, dan segera membalas klien. Ketika pembaruan papan pemimpin selesai, backend dapat mengirim pembaruan ke klien game sehingga tampilan pemain tentang papan pemimpin diperbarui.

Atau, pemain cukup mengunjungi layar papan pemimpin game Anda untuk mengambil data terbaru, yang dapat mengeluarkan permintaan web ke backend Anda untuk mengambil data terbaru dari cache.

Langkah-langkah implementasi

- Tentukan apakah interaksi client-backend memerlukan komunikasi sinkron; terapkan pendekatan asinkron dan non-pemblokiran jika memungkinkan untuk mengoptimalkan penggunaan sumber daya.
- Gunakan Amazon SQS untuk membongkar tugas yang tidak penting seperti pembaruan papan peringkat.
- Izinkan klien untuk mengambil data yang diperbarui secara asinkron, seperti mengambil data papan peringkat terbaru sesuai permintaan atau melalui pembaruan latar belakang.

Sumber daya

- Memahami pesan asinkron untuk layanan mikro
- Lambda - Menggunakan integrasi layanan dan pemrosesan asinkron

## Manajemen data

GAMEPERF06: Bagaimana Anda mengelola dan menganalisis log server game secara efisien dan menyimpan berbagai jenis data game untuk kinerja yang optimal?

Game dapat memiliki data pemain, log game, dan log server yang harus dipisahkan satu sama lain jika memungkinkan. Memusatkan konsumsi log dan manajemen siklus hidup dapat menguntungkan tim game Anda dengan memberikan wawasan tentang apa yang terjadi di game dan di server.

Praktik terbaik

- GAMEPERF06-BP01 Memusatkan pengumpulan dan penyimpanan log
- GAMEPERF06-BP02 Mengkategorikan dan menyimpan data game berdasarkan pola akses

- GAMEPERF06-BP03 Aktifkan pemformatan dan pengelompokan log yang efisien
- GAMEPERF06-BP04 Menerapkan kebijakan rotasi dan retensi log
- GAMEPERF06-BP05 Gunakan alat pemantauan dan visualisasi

## GAMEPERF06-BP01 Memusatkan pengumpulan dan penyimpanan log

Menerapkan koleksi log terpusat dan solusi penyimpanan untuk mengumpulkan log dari instance server game dan. GameLift

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Gunakan layanan seperti Amazon CloudWatch Logs untuk mengumpulkan, memantau, dan menyimpan data log dari server dan GameLift instance game Anda. CloudWatch Log menyediakan solusi yang dapat diskalakan dan dikelola sepenuhnya untuk manajemen log, memfasilitasi penyimpanan dan pengambilan data log yang efisien tanpa memengaruhi kinerja server game. Jika Anda menjalankan agen CloudWatch Log, pertimbangkan berbagai jenis instalasi dan opsi konfigurasi seperti ukuran batch, durasi buffer untuk meminimalkan dampak ke server game. Pertimbangkan instance server game sementara dan kurangi ketergantungan pada logging lokal jika memungkinkan. Menetapkan kebijakan terpusat untuk implementasi praktik terbaik Logging.

Langkah-langkah implementasi

- Gunakan CloudWatch Log Amazon untuk mengumpulkan, memantau, dan menyimpan data log dari instance server game dan GameLift, memfasilitasi manajemen log terpusat dan terukur.

## GAMEPERF06-BP02 Mengkategorikan dan menyimpan data game berdasarkan pola akses

Kategorikan data game Anda ke dalam berbagai jenis berdasarkan pola akses dan persyaratan penyimpanannya.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Kategori umum termasuk data pemain, penyimpanan game, penyimpanan dunia persisten, dan data analitik.

Langkah-langkah implementasi

Gunakan solusi penyimpanan yang tepat untuk setiap tipe data untuk mengoptimalkan kinerja dan efisiensi biaya:

- Data pemain: Gunakan Amazon DynamoDB, database NoSQL yang cepat dan dapat diskalakan, untuk menyimpan profil pemain, preferensi, dan data perkembangan. Akses latensi rendah dan kemampuan penskalaan otomatis DynamoDB memberikan pengambilan dan pembaruan data pemain yang efisien.
- Penyimpanan game: Gunakan Amazon S3 untuk menyimpan penyimpanan game dan pos pemeriksaan. S3 memberikan daya tahan dan skalabilitas tinggi untuk menyimpan data penyimpanan game dalam jumlah besar. Pertimbangkan untuk menggunakan S3 Transfer Acceleration atau Amazon CloudFront untuk mengunggah dan mengunduh penyimpanan game yang lebih cepat.
- Penyimpanan dunia persisten: Untuk game dengan status dunia persisten atau data game bersama, pertimbangkan untuk menggunakan Amazon DynamoDB, Amazon, atau ElastiCache Amazon MemoryDB. ElastiCache dan MemoryDB menyediakan penyimpanan nilai kunci dalam memori sementara DynamoDB adalah database NoSQL yang didukung SSD. Layanan ini menyediakan akses cepat ke data yang disimpan, mengurangi waktu yang diperlukan untuk proses server game untuk menyimpan status game yang meningkatkan kinerja proses secara keseluruhan.
- Data analitik: Gunakan Amazon Managed Streaming for Apache Kafka atau Kinesis Data Streams untuk menyerap aliran data dari produsen data game Anda. Amazon Managed Service untuk Apache Flink dapat digunakan untuk transformasi dan analisis real-time dan dikirim ke Amazon Data Firehose untuk diproses dan dikirim ke danau data backend, gudang, dan layanan analitik. Panduan untuk Game Analytics Pipeline tentang AWS menggambarkan bagaimana layanan bekerja sama untuk menyediakan analisis hampir real-time dan batch.

## GAMEPERF06-BP03 Aktifkan pemformatan dan pengelompokan log yang efisien

Konfigurasikan proses server game Anda untuk menghasilkan log dalam terstruktur dan dalam format yang dapat diuraikan, seperti JSON.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Terapkan teknik pengelompokan log untuk meminimalkan frekuensi transfer data log dari server game Anda ke penyimpanan log terpusat. Batching log mengurangi overhead jaringan dan meningkatkan kinerja server game. Gunakan log tingkat verbose atau debug sebagai pengecualian dan bukan default, karena mereka dapat dikenakan penalti kinerja dan biaya yang harus dihindari jika memungkinkan.

## GAMEPERF06-BP04 Menerapkan kebijakan rotasi dan retensi log

Menetapkan kebijakan rotasi dan retensi log untuk mengelola pertumbuhan data log dan mengoptimalkan pemanfaatan penyimpanan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

### Panduan implementasi

Konfigurasikan server game Anda untuk memutar log secara otomatis berdasarkan ukuran atau interval waktu. Tentukan kebijakan penyimpanan log di Amazon CloudWatch Logs untuk mengarsipkan atau menghapus data log lama secara otomatis yang tidak lagi diperlukan untuk analisis aktif atau pemecahan masalah.

## GAMEPERF06-BP05 Gunakan alat pemantauan dan visualisasi

Gunakan alat pemantauan dan visualisasi untuk mendapatkan wawasan tentang kinerja server game Anda dan mengidentifikasi peluang pengoptimalan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Gunakan Amazon CloudWatch untuk memantau metrik kunci dan mengatur alarm untuk notifikasi proaktif. Manfaatkan alat seperti Amazon Managed Service untuk Prometheus dan Amazon Managed Grafana untuk mengumpulkan, menanyakan, dan memvisualisasikan metrik dari server dan infrastruktur game Anda. Buat dasbor informatif untuk melacak kinerja, mengidentifikasi kemacetan, dan membuat pengoptimalan berbasis data.

# Jaringan dan Pengiriman Konten

GAMEPERF07: Bagaimana Anda mendesain layanan perjodohan Anda untuk mengoptimalkan kinerja?

Keterampilan pemain, kualitas penyedia layanan internet (ISP), dan distribusi populasi pemain penting untuk dipertimbangkan sebagai dimensi penyetelan kinerja. Sesi permainan dapat ditempatkan di server yang berlokasi strategis untuk meratakan lapangan bermain dan menyelenggarakan permainan yang adil.

Praktik terbaik

- GAMEPERF07-BP01 Tentukan ambang batas latensi jaringan untuk game Anda
- GAMEPERF07-BP02 Jalankan layanan perjodohan terpisah untuk setiap mode gameplay dan wilayah hosting game
- GAMEPERF07-BP03 Secara teratur memantau kinerja perjodohan
- GAMEPERF07-BP04 Secara teratur memantau kinerja jaringan
- GAMEPERF07-BP05 Gunakan teknologi akselerasi jaringan untuk meningkatkan kinerja di internet

## GAMEPERF07-BP01 Tentukan ambang batas latensi jaringan untuk game Anda

Saat mengembangkan game multipemain, verifikasi bahwa infrastruktur game Anda tidak memperkenalkan latensi yang tidak perlu bagi pemain. Jika game Anda sensitif terhadap latensi jaringan, maka Anda harus menetapkan ambang batas latensi dalam logika perjodohan Anda untuk memprioritaskan penempatan pemain pada sesi server game yang di-host di lokasi server game terdekat atau yang memenuhi tujuan Anda untuk pengalaman pemain Wilayah AWS yang ideal.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Dalam banyak game yang sensitif terhadap latensi, adalah umum untuk menginstruksikan klien game untuk melakukan ping ke setiap lokasi infrastruktur game untuk mengumpulkan data kinerja seperti latensi jaringan, jitter, dan kehilangan paket, dan melaporkan data ini ke backend pengumpulan

metrik sehingga dapat dianalisis. Saat mencocokkan pemain ke dalam sesi game, Anda dapat mengonfigurasi game Anda untuk menggabungkan latensi jaringan yang dirasakan klien game ke infrastruktur server game Anda sebagai salah satu input yang digunakan dalam logika layanan perjodohan Anda.

## GAMEPERF07-BP02 Jalankan layanan perjodohan terpisah untuk setiap mode gameplay dan wilayah hosting game

Jika gim Anda menawarkan beberapa mode gameplay untuk dipilih pemain, Anda harus memisahkan sistem perjodohan untuk masing-masing mode sehingga Anda dapat secara mandiri menyetel kinerja untuk setiap mode gameplay berdasarkan persyaratan uniknya dan mengurangi pertengkaran sumber daya. Setiap mode gameplay kemungkinan akan memiliki persyaratan unik untuk latensi yang dapat diterima, ukuran kecocokan, serta logika perjodohan khusus permainan khusus lainnya. Mereka juga kemungkinan akan menarik berbagai jenis pemain. Jalankan setiap layanan perjodohan mode permainan sebagai penyebaran perangkat lunak terpisah sehingga Anda dapat menguji kinerja dan mengoperasikan mode permainan secara mandiri.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Misalnya, Anda dapat menjalankannya sebagai fungsi Lambda terpisah untuk setiap mode permainan, atau Anda dapat mengoperasikannya sebagai penerapan layanan berbasis kontainer yang terpisah.

Terapkan layanan perjodohan Anda ke beberapa Wilayah di dekat lokasi server game Anda. Lalu lintas pemain akan mengambil banyak rute, jadi penting bagi layanan perjodohan untuk mempertahankan profil up-to-date latensi di beberapa rute ISPs untuk meningkatkan efisiensi penempatan sesi permainan latensi rendah. GameLift FlexMatch memberikan panduan tambahan untuk memilih Wilayah untuk mak comblang, dan mencakup kemampuan untuk mengintegrasikan mak comblang Anda dengan antrian sesi permainan Multi-wilayah.

## GAMEPERF07-BP03 Secara teratur memantau kinerja perjodohan

Salah satu cara paling mencolok untuk mengoptimalkan kinerja permainan bagi pemain adalah dengan mengurangi waktu yang harus mereka tunggu sebelum mereka dapat memasuki sesi permainan. Waktu tunggu yang lama dapat menyebabkan pemain kehilangan minat dan menyebabkan gesekan, jadi penting untuk mempertimbangkan hal ini saat merancang solusi perjodohan Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Saat Anda merancang konfigurasi perjodohan untuk permainan Anda, buat aturan yang menentukan kondisi yang diterapkan untuk membentuk kecocokan. Anda harus mempertimbangkan dampak aturan ini terhadap kinerja sistem, terutama waktu tunggu untuk pemain. Sebelum menerapkan perubahan pada implementasi perjodohan Anda, seperti penambahan kondisi atau filter perjodohan baru, uji ini sebelumnya atau pertimbangkan untuk merilis perubahan ini secara bertahap ke populasi sampel kecil pemain sebagai kenari atau A/B uji untuk mengumpulkan metrik kinerja sebelum memperkenalkan perubahan ini ke seluruh populasi pemain.

Konfigurasikan layanan perjodohan Anda untuk menghasilkan log terperinci guna memahami kondisi atau aturan yang diterapkan pada setiap permintaan perjodohan. Ini membantu dalam tinjauan dan menyesuaikan implementasi perjodohan seperlunya.

Misalnya, Amazon GameLift FlexMatch menyediakan layanan perjodohan yang dikelola sepenuhnya yang dapat digunakan sebagai layanan mandiri dengan hosting server game Anda sendiri atau digunakan dengan server game yang dihosting di Amazon. GameLift FlexMatch dapat menghasilkan pemberitahuan acara ke Amazon EventBridge, lihat Mengatur pemberitahuan FlexMatch acara. Gunakan Amazon Simple Notification Service (Amazon SNS) untuk menerima data perjodohan dalam format JSON, memungkinkan Anda memproses dan menyimpan informasi ini secara otomatis untuk analisis guna meningkatkan kinerja perjodohan.

Siapkan metrik untuk melacak berapa lama layanan perjodohan Anda untuk menemukan sesi permainan yang cocok untuk pemain. Tinjau metrik durasi perjodohan secara teratur dan korelasikan waktu-waktu ini dengan perilaku pemain dan sentimen komunitas. Gunakan data ini untuk mengembangkan ambang batas yang sesuai untuk batas waktu perjodohan yang dapat disertakan dalam konfigurasi aturan perjodohan Anda.

Misalnya, Amazon GameLift FlexMatch menyediakan dukungan untuk menentukan batas waktu permintaan perjodohan serta membuat aturan perjodohan yang dapat memungkinkan persyaratan untuk bersantai dari waktu ke waktu. Fitur ini memungkinkan Anda untuk membuat perjodohan yang dapat beradaptasi untuk membuatnya mudah untuk membuat pertandingan dan menempatkan pemain ke dalam sesi permainan ketika pertandingan sulit ditemukan.

## GAMEPERF07-BP04 Secara teratur memantau kinerja jaringan

Untuk permainan kompetitif, penting untuk memiliki pengalaman pemain yang konsisten.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Sebuah permainan yang andal 50ms untuk basis pemain yang lebih besar lebih adil dan lebih menyenangkan daripada pertandingan di mana satu pemain memiliki ping 10 ms dan yang lain yang memiliki ping 70 ms. Perubahan perutean ISP dapat memengaruhi sebagian populasi pemain, dan sistem perjodohan Anda perlu beradaptasi. Amazon CloudWatch Network Monitoring membantu menentukan apakah masalahnya ada pada game Anda atau penyedia internet pemain.

Langkah-langkah implementasi

- Gunakan Amazon Cloudwatch Network Monitoring untuk melacak kinerja jaringan dan mengidentifikasi masalah perutean.
- Gunakan VPC Flow Logs untuk mengidentifikasi pola lalu lintas abnormal atau paket yang terjatuh, yang dapat menunjukkan kemacetan jaringan, masalah ISP, atau kesalahan konfigurasi yang memengaruhi latensi pemain.

## GAMEPERF07-BP05 Gunakan teknologi akselerasi jaringan untuk meningkatkan kinerja di internet

Selain menempatkan infrastruktur game yang sensitif terhadap latensi secara fisik lebih dekat dengan pemain, Anda juga dapat meningkatkan pengalaman pemain dengan mengoptimalkan kinerja jaringan untuk game Anda. AWS menggunakan protokol BGP untuk mempengaruhi perutean internet untuk menggunakan jalur tercepat ke jaringan perbatasan kami dari Penyedia Layanan Internet. Jika Anda mengoperasikan jaringan Anda sendiri dan membutuhkan lebih banyak kontrol dan pengamatan atas perilaku perutean dan iklan BGP, Anda dapat menggunakan Peering pribadi atau Direct Connect untuk mengarahkan lalu lintas dari internet ke permainan Anda yang sedang berjalan. AWS

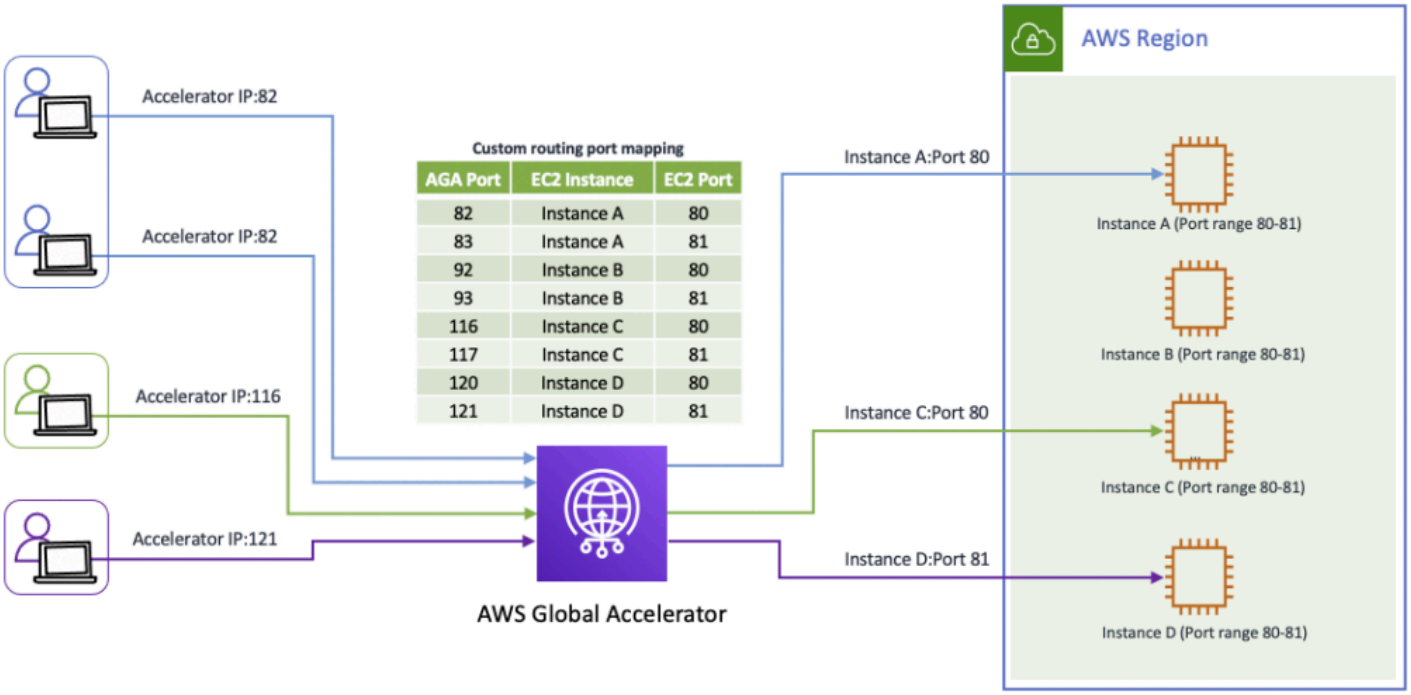
Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Pertimbangkan arsitektur referensi berikut untuk mendukung peningkatan kinerja dan daya tanggap internet.

Peningkatan kinerja jaringan untuk bermain game menggunakan Global Accelerator

Untuk solusi yang dikelola sepenuhnya untuk perutean jaringan, AWS Global Accelerator meningkatkan kinerja jaringan aplikasi Anda menggunakan jaringan AWS global, yang dapat digunakan untuk mempercepat lalu lintas gameplay, obrolan suara, dan lalu lintas pesan real-time, serta aplikasi sensitif latensi lainnya sambil memberikan failover cepat ke server game Anda. Akselerator perutean kustom Global Accelerator dapat diintegrasikan dengan layanan perjodohan Anda untuk menyediakan perutean deterministik beberapa pemain ke sesi permainan yang sama menggunakan alamat IP dan port anycast statis.




Tim pengembangan game Anda dapat didistribusikan ke seluruh dunia dan memerlukan akses kinerja ke konten atau aset bersama. Untuk meningkatkan kinerja konten bersama yang disimpan di bucket Amazon S3, Anda dapat mengatur replikasi dua arah data Anda di seluruh Wilayah menggunakan Replikasi Lintas Wilayah S3 sehingga pengguna dapat mengakses data dari bucket yang lebih dekat dengannya. Untuk menyederhanakan pola akses ini, gunakan S3 Multi-Region Access Points yang mempercepat permintaan ke S3 melalui jaringan global menggunakan Global Accelerator.

Untuk informasi selengkapnya, lihat Meningkatkan Pengalaman Pemain dengan Memanfaatkan Akselerator AWS Global dan Amazon GameLift FleetiQ.

Langkah-langkah implementasi

- Gunakan AWS Global Accelerator untuk membantu meningkatkan kinerja jaringan untuk lalu lintas gameplay, obrolan suara, dan pesan waktu nyata, sambil memfasilitasi failover cepat ke server game.
- Konfigurasikan akselerator perutean kustom Global Accelerator untuk diintegrasikan dengan layanan perjodohan Anda, memungkinkan perutean deterministik pemain ke sesi game menggunakan anycast statis. IPs
- Aktifkan Replikasi Lintas Wilayah S3 untuk mereplikasi konten bersama di seluruh Wilayah untuk tim pengembangan game terdistribusi.
- Gunakan S3 Multi-Region Access Points untuk mempercepat akses data S3 melalui jaringan AWS global untuk pengguna yang didistribusikan secara global.

# Proses dan budaya

GAMEPERF08: Bagaimana Anda menyelaraskan bilah kinerja untuk game Anda dengan harapan pemain dan pengembang?

Mengetahui pemain dan pengembang Anda adalah salah satu aspek terpenting dalam meningkatkan efisiensi kinerja. Menyampaikan game berkinerja dengan overhead operasional rendah adalah salah satu cara terbaik untuk menunjukkan kepada pemain dan pengembang bahwa Anda peduli dengan pengalaman mereka dan dapat membedakan game dan studio Anda.

Praktik terbaik

- GAMEPERF08-BP01 Menginformasikan dan menyertakan pemain dalam proses Anda
- GAMEPERF08-BP02 Sejajarkan pemilihan solusi dengan keterampilan dan keahlian tim teknik

## GAMEPERF08-BP01 Menginformasikan dan menyertakan pemain dalam proses Anda

Berikan opsi untuk ditampilkan dalam metrik game seperti latensi, frame per detik, dan paket yang dijatuhkan. Masalah infrastruktur permukaan dan downtime pemeliharaan melalui komunikasi yang

dihadapi pemain seperti halaman status. Rayakan lokasi permainan baru dengan komunikasi pemain termasuk blog dev dan tetapkan harapan untuk peningkatan pengalaman pemain yang diharapkan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

### Sertakan pemain

Berikan proses pengiriman diagnostik sederhana yang mengumpulkan file yang relevan dan menempelkannya ke tiket dukungan pemain dari klien game Anda. Aktifkan forum dukungan di mana pemain dapat saling membantu dan menjadi bagian dari meningkatkan pengalaman bermain game

### Pertimbangkan trade-off versus ekspektasi pemain

Memindahkan sistem backend untuk efisiensi biaya mungkin tidak terlihat oleh pemain tetapi server game yang bergerak dapat mengubah waktu ping. Bersikaplah konsisten dan adil kepada pemain dengan alasan untuk ekspansi dan pengurangan lokasi hosting game Anda.

Komunitas pemain dan geografi akan memiliki karakteristik mereka sendiri yang dapat memengaruhi harapan permainan Anda. Misalnya, Korea Selatan memiliki beberapa internet tercepat di planet ini dan harapan untuk gameplay adalah latensi satu digit yang mendorong permainan yang sangat kompetitif. Gameplay kasual di perangkat seluler menciptakan profil kinerja dan pola penggunaan yang berbeda dibandingkan dengan permainan sesi konsol dan PC.

Login dan lobi adalah bagian dari pengalaman dan harus merasa responsif, bahkan jika server sedang offline untuk pemeliharaan. Raid night planning atau nongkrong di lobi adalah bagian dari pengalaman pemain dan penting untuk dipertimbangkan ketika memilih area fokus untuk efisiensi kinerja. Pemain dapat membiarkan klien game Anda terbuka selama berbulan-bulan, kadang-kadang mereka mungkin hanya masuk sesekali untuk membaca catatan tambalan. Game Live Ops perlu mengingat seluruh pengalaman pemain sebagai bagian dari proses dan budaya rekayasa.

### Langkah-langkah implementasi

- Berikan metrik dalam game seperti latensi, FPS, dan kehilangan paket, dan komunikasikan masalah infrastruktur dan jadwal pemeliharaan melalui halaman status dan pembaruan yang dihadapi pemain.
- Menerapkan fitur dump dan submission diagnostik di klien game dan buat forum dukungan untuk mendorong pemecahan masalah dan peningkatan berbasis komunitas.

- Sesuaikan optimasi kinerja dengan ekspektasi komunitas pemain, seperti latensi rendah untuk Wilayah kompetitif atau login/lobby pengalaman responsif untuk pemain kasual dan sesi panjang.
- Rancang alur kerja Live Ops untuk memperhitungkan seluruh pengalaman pemain, mulai dari gameplay aktif hingga perilaku klien yang menganggur, memfasilitasi keterlibatan tanpa batas.

## GAMEPERF08-BP02 Sejajarkan pemilihan solusi dengan keterampilan dan keahlian tim teknik

Nilai keterampilan dan keahlian tim Anda dalam mengelola dan mengoptimalkan kinerja server game saat memilih opsi hosting Anda. Solusi yang dihosting sendiri seperti EC2 dan kontainer membutuhkan lebih banyak pengetahuan tentang manajemen infrastruktur, penyetelan kinerja, dan penskalaan. Jika tim Anda tidak memiliki keterampilan ini, layanan terkelola seperti GameLift mungkin lebih cocok, karena mengabstraksi banyak kompleksitas dan memungkinkan tim Anda untuk fokus pada pengoptimalan khusus permainan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Dengan mengevaluasi faktor-faktor ini dan melakukan tes kinerja di berbagai opsi hosting, Anda dapat memilih solusi yang paling tepat yang memenuhi persyaratan spesifik game Anda sambil mengoptimalkan efisiensi kinerja.

# Sumber daya

Pelajari lebih lanjut tentang praktik terbaik kami terkait efisiensi kinerja.

Dokumen terkait:

- Pusat Arsitektur AWS
- Pilar Efisiensi Kinerja - Kerangka AWS Well-Architected
- Membandingkan pola penyimpanan lokal Anda dengan layanan AWS Penyimpanan
- Instans menyimpan penyimpanan blok sementara untuk EC2 instance
- Kumpulkan metrik, log, dan jejak menggunakan agen CloudWatch
- CloudWatchAgen
- Bagaimana cara mengaktifkan dan mengonfigurasi jaringan yang disempurnakan pada EC2 instans saya?

- Meningkatkan Pengalaman Pemain dengan Memanfaatkan AWS Global Accelerator dan Amazon FleetiQ GameLift
- Blog Teknologi Riot Games: Skalabilitas dan Pengujian Beban Untuk Valorant
- Game online hiper-skala dengan Solusi hybrid AWS
- Metode Pemanfaatan Saturasi dan Kesalahan (USE)
- Instans EC2 Spot Amazon
- Kinerja dalam Skala dengan Amazon ElastiCache
- Strategi Caching Database Menggunakan Redis
- Pilihan Konektivitas Amazon Virtual Private Cloud
- Pola Desain Praktik Terbaik: Mengoptimalkan Kinerja Amazon S3

Tolok ukur terkait:

- Benchmark volume Amazon EBS
- Bandwidth jaringan instans Amazon EC@

Alat terkait:

- Unreal Engine: Menguji dan Mengoptimalkan Konten Anda
- Profiler Persatuan
- Sistem Kualitas Mesin 3D Terbuka (O3DE)
- Memantau GameLift Server Amazon
- Toolkit GameLift Pengujian Amazon

Video terkait:

- AWS re: ciptakan 2019: [ULANGI 2] EC2 Yayasan Amazon (-R2) CMP211
- AWS Re:invent 2019: Memberdayakan EC2 Amazon generasi berikutnya: Menyelam jauh ke dalam sistem Nitro (03-R2) CMP3
- Memulai Amazon GameLift FleetiQ AWS - Online Tech Talks
- Zach Blitz dari Riot Games tentang Penggunaan untuk Meningkatkan Permainan AWS
- AWS re: invent 2023 - AWS Graviton: Kinerja harga terbaik untuk beban kerja Anda () AWS CMP334

Pelatihan terkait:

- Hosting Server Game di AWS
- Menggunakan Amazon GameLiftFleet IQ untuk Server Game
- Memulai AWS untuk Game — Bagian I
- Hosting Server Game di AWS
- AWS re: Invent 2023 - AWS Graviton: Kinerja harga terbaik untuk beban kerja Anda () AWS CMP334

# Optimalisasi biaya

Pilar optimasi biaya mencakup proses penyempurnaan dan peningkatan sistem yang berkelanjutan selama seluruh siklus hidupnya. Proses ini terbentang dari desain awal bukti konsep pertama Anda hingga pengoperasian beban kerja produksi yang sedang berlangsung. Dengan mengadopsi garis besar praktik dalam paper ini, Anda dapat membangun dan mengoperasikan sistem sadar biaya yang mencapai hasil bisnis yang diinginkan pada titik harga terendah. Menerapkan praktik pengoptimalan biaya ini dapat memungkinkan bisnis Anda memaksimalkan nilai investasi cloud Anda.

Game adalah proyek kreatif unik yang harus bersaing untuk perhatian pemain dan waktu bermain. Sebelum diluncurkan, pengembang game sering tidak memiliki pemahaman yang jelas tentang seberapa populer atau tahan lama game mereka nantinya. Bergantung pada strategi monetisasi game, prioritas bisnis, dan tahap siklus hidup, pengembang perlu melakukan pengorbanan saat mengevaluasi keputusan pengoptimalan biaya.

Misalnya, selama fase pra-peluncuran game baru yang sangat dinanti, fokusnya biasanya pada speed-to-market, pengembangan fitur, dan kinerja. Prioritasnya adalah memverifikasi skala infrastruktur untuk memenuhi permintaan pemain puncak. Sebaliknya, jika permainan tidak berhasil atau pengembangan melambat, fokusnya dapat beralih ke pengurangan biaya sebanyak mungkin untuk terus mengoperasikan game untuk pemain yang ada.

Banyak pengembang game juga mengoperasikan beberapa game secara bersamaan, yang membutuhkan pertimbangan tambahan. Sumber daya seperti infrastruktur, perangkat lunak, dan staf dapat dibagi di beberapa game langsung, memungkinkan kerugian dari satu game diimbangi oleh keuntungan dari yang lain. Dalam skenario ini, fokus pada optimasi biaya dapat meningkatkan keuangan di seluruh portofolio game.

Mengingat model bisnis yang unik, skala, dan ketidakpastian permainan, pertanyaan kunci berikut dapat memandu keputusan pengoptimalan biaya:

- Bagaimana cara mengukur biaya infrastruktur per pemain, sistem, dan fitur game?
- Apa keseimbangan yang tepat antara pengoptimalan biaya dan pengalaman pemain untuk tahap siklus hidup game saya saat ini?
- Bagaimana saya bisa memaksimalkan laba atas investasi menggunakan model penetapan harga yang tepat untuk AWS sumber daya saya?

Menerapkan praktik terbaik ini dan mengajukan pertanyaan yang tepat dapat membantu pengembang game membangun dan mengoperasikan sistem sadar biaya yang mencapai hasil bisnis sambil meminimalkan biaya.

Area fokus

- Prinsip desain
- Mempraktikkan Manajemen Keuangan Cloud
- Kesadaran akan penggunaan dan pengeluaran
- Sumber daya hemat biaya
- Biaya transfer data
- Mengelola permintaan dan sumber daya pasokan
- Mengoptimalkan dari waktu ke waktu
- Sumber daya

## Prinsip desain

Selain prinsip-prinsip desain dari pilar optimasi biaya dari Well-Architected Framework, prinsip-prinsip desain berikut mengoptimalkan biaya menjalankan beban kerja game Anda di cloud.

- Ukur biaya infrastruktur per pemain, sistem, dan fitur game: Memahami dan melacak biaya infrastruktur yang diperlukan untuk pengalaman dan fitur pemain tertentu di seluruh sistem game. Ini dapat mengidentifikasi area arsitektur Anda yang mungkin memerlukan pengoptimalan biaya.
- Nilai trade-off pengoptimalan biaya versus pengalaman pemain: Evaluasi tahap permainan Anda untuk menentukan fokus yang tepat - pengalaman pemain atau pengoptimalan biaya. Biasanya, begitu permainan mencapai massa kritis dan populasi pemain stabil, sekarang saatnya untuk fokus pada pengoptimalan biaya operasi. Kuncinya adalah menyeimbangkan memberikan pengalaman pemain hebat dengan menjalankan infrastruktur game Anda dengan cara yang paling hemat biaya. Menerapkan prinsip-prinsip desain ini memaksimalkan laba atas investasi game Anda.

## Mempraktikkan Manajemen Keuangan Cloud

Tidak ada praktik terbaik Cloud Financial Management khusus untuk Game Lens. Lihat Pilar Pengoptimalan Biaya Kerangka Well-Architected Framework untuk panduan tentang manajemen keuangan cloud.

# Kesadaran akan penggunaan dan pengeluaran

GAMECOST01: Bagaimana Anda mengukur biaya lingkungan game Anda?

Pahami biaya per pemain, fitur game, dan lingkungan sehingga Anda dapat mengelola dan memperkirakan pengeluaran Anda karena jumlah pemain berubah seiring waktu dan fitur ditambahkan dan ditingkatkan. Pertimbangkan praktik terbaik berikut untuk mengelola biaya lingkungan permainan Anda yang berbeda.

Praktik terbaik

- GAMECOST01-BP01 Menerapkan atribusi biaya per pemain, fitur game, dan lingkungan
- GAMECOST01-BP02 Temukan peluang untuk optimasi

## GAMECOST01-BP01 Menerapkan atribusi biaya per pemain, fitur game, dan lingkungan

Atribusi biaya untuk server game biasanya lebih mudah dilakukan daripada layanan backend game karena server game biasanya dioptimalkan untuk dapat meng-host sejumlah pemain bersamaan per instance yang dapat diamortisasi di seluruh biaya menjalankan instance.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Untuk layanan backend game, disarankan untuk menghapus komponen permainan Anda ke dalam fitur berbeda yang dapat dikelola sebagai sumber daya logis atau fisik terpisah untuk membuatnya mudah untuk menganalisis biaya.

Misalnya, meskipun mungkin tampak mudah untuk menerapkan aplikasi monolitik tunggal untuk meng-host layanan backend game, pola ini membuat sulit untuk mendapatkan total biaya per pemain dan fitur game dari waktu ke waktu saat Anda menambahkan lebih banyak fitur karena komputasi, jaringan, dan biaya penyimpanan sumber daya dibagi di seluruh fitur. Pertimbangkan untuk mengadopsi arsitektur tanpa server untuk layanan backend game Anda dengan layanan seperti Amazon API Gateway dan AWS Lambda atau untuk AWS Fargate komputasi, Amazon SQS dan Amazon SNS untuk perpesanan, Amazon S3 untuk penyimpanan objek, dan Amazon

DynamoDB untuk penyimpanan database. Layanan ini hanyalah beberapa contoh produk yang menawarkan harga yang berbasis penggunaan dan terutama didorong oleh volume permintaan sehingga biaya dapat divisualisasikan dengan perincian. Sumber daya individual seperti fungsi Lambda, layanan Fargate, tabel DynamoDB, dan bucket S3 dapat dikaitkan dengan tag alokasi biaya sehingga Anda dapat menghubungkan biaya layanan ini dengan nama fitur game yang membuatnya mudah bagi Anda untuk memahami biaya untuk masing-masing layanan Anda.

Disarankan juga untuk mengelola secara terpisah setiap lingkungan pengembangan game Anda sehingga Anda dapat mengaitkan biaya untuk lingkungan yang berbeda. Biasanya, pengembang game akan mengelola lingkungan terpisah untuk pengembangan, pengujian, pementasan, dan lingkungan produksi, seperti yang dijelaskan dalam pilar operasi lensa industri game ini. Setiap lingkungan biasanya memiliki skalabilitas, kinerja, dan persyaratan penggunaan yang berbeda dan dapat dikelola oleh tim yang terpisah. Untuk mengontrol biaya, atur lingkungan ini sehingga Anda dapat memantau dan mengaitkan biaya setiap lingkungan dengan benar.

Untuk informasi lebih lanjut, lihat dokumentasi berikut:

- Membangun game multi-pemain tanpa server yang berskala
- Server sesi game mandiri dengan backend WebSockets berbasis
- Server sesi game mandiri dengan backend tanpa server

Langkah-langkah implementasi

- De-couple layanan backend game ke dalam fitur berbeda menggunakan arsitektur tanpa server atau kontainer seperti, Amazon AWS Lambda API Gateway, AWS Fargate dan untuk mengaktifkan atribusi biaya granular per fitur.
- Terapkan tag alokasi biaya ke sumber daya individu (misalnya, fungsi Lambda, tabel DynamoDB, dan bucket S3) untuk mengaitkan biaya dengan fitur game tertentu untuk analisis biaya yang lebih baik.
- Mengelola lingkungan terpisah untuk pengembangan, pengujian, pementasan, dan produksi, mengatur dan memantau biaya mereka secara independen untuk menyelaraskan dengan skalabilitas dan persyaratan penggunaan.

## GAMECOST01-BP02 Temukan peluang untuk optimasi

Pengembang dan penerbit game dapat menggunakan AWS FinOps praktik untuk membantu mengoptimalkan biaya cloud mereka dan mendapatkan visibilitas yang lebih baik ke dalam

pengeluaran cloud mereka. Dengan demikian, produsen game dapat menyelaraskan biaya rata-rata yang diperlukan untuk memelihara infrastruktur bagi para pemain dengan hasil keuangan yang disampaikan oleh game.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

## Panduan implementasi

AWS menawarkan panduan solusi siap pakai untuk Cloud Financial Management untuk mengelola dan mengoptimalkan pengeluaran Anda untuk layanan cloud. Kemampuan ini mencakup visibilitas granular dan analisis biaya dan penggunaan untuk mendukung pengambilan keputusan untuk topik seperti dasbor pengeluaran, pengoptimalan, batas pengeluaran, pengisian kembali, dan deteksi dan respons anomali. Panduan solusi untuk Cloud Financial Management mencakup fitur anggaran dan perkiraan, memberi Anda arsitektur yang ditentukan dan dioptimalkan biaya untuk beban kerja Anda sehingga Anda dapat memilih model penetapan harga yang tepat dan atribut biaya sumber daya yang relevan dengan tim Anda. Ini mengaktifkan teknik pelacakan, pemberitahuan, dan pengoptimalan biaya di seluruh lingkungan dan sumber daya Anda. Anda dapat mengelola informasi pengeluaran secara terpusat dan memberikan akses kepada pemangku kepentingan penting sesuai kebutuhan untuk visibilitas yang ditargetkan dan untuk mendukung pengambilan keputusan.

FinOps Alat utama lainnya adalah Hub Pengoptimalan Biaya, yang memberikan pandangan terpusat tentang rekomendasi dan peluang pengoptimalan biaya di seluruh Anda Akun AWS dan Wilayah AWS, sehingga Anda bisa mendapatkan hasil maksimal dari AWS pengeluaran Anda. Anda dapat menggunakan Cost Optimization Hub untuk mengidentifikasi, memfilter, dan mengumpulkan rekomendasi pengoptimalan AWS biaya di seluruh area Anda Akun AWS dan Wilayah AWS. Itu membuat rekomendasi tentang hak sumber daya, penghapusan sumber daya yang tidak digunakan, Savings Plans, dan Instans Cadangan. Dengan satu dasbor, Anda menghindari keharusan pergi ke beberapa AWS produk untuk mengidentifikasi peluang pengoptimalan biaya.

Jika tim game Anda menggunakan MyApplications bersama Akun AWS di Konsol Manajemen AWS Beranda, dapat digunakan untuk melihat biaya sumber daya aplikasi untuk beban kerja individual. Tampilan terperinci ini memungkinkan Anda mengidentifikasi tren biaya spesifik dalam infrastruktur game Anda, memungkinkan Anda membuat keputusan berdasarkan informasi tentang alokasi dan pengoptimalan sumber daya.

Selain itu, secara teratur meninjau data penagihan dan manajemen biaya Anda dengan AWS Data Exports mengungkap peluang penghematan biaya tersembunyi. Laporan terperinci ini memberikan rincian komprehensif tentang pengeluaran cloud Anda, memungkinkan Anda mengidentifikasi area

pengeluaran berlebihan, sumber daya yang tidak digunakan, dan peluang untuk memanfaatkan layanan atau model penetapan harga yang lebih hemat biaya.

Dengan merangkul FinOps prinsip dan memanfaatkan alat yang disediakan oleh AWS, pengembang dan penerbit game dapat memanfaatkan sumber daya cloud mereka dengan seefisien mungkin, pada akhirnya meningkatkan laba mereka dan membebaskan dana untuk pengembangan dan inovasi game lebih lanjut.

Langkah-langkah implementasi

- Gunakan alat Manajemen AWS Cloud Keuangan untuk visibilitas terperinci dan terperinci, dasbor pengeluaran, deteksi anomali, dan atribusi biaya untuk mengoptimalkan dan melacak pengeluaran cloud secara efektif.
- Gunakan Cost Optimization Hub untuk memusatkan rekomendasi rightsizing, Savings Plans, dan Reserved Instance di seluruh dan Wilayah. Akun AWS
- Tinjau data AWS penagihan secara teratur menggunakan Ekspor Data dan MyApplication seterusnya AWS untuk membantu menganalisis biaya spesifik beban kerja, mengungkap peluang penghematan, dan mengoptimalkan alokasi sumber daya.

## Sumber daya hemat biaya

GAMECOST02: Bagaimana Anda memilih solusi komputasi yang tepat untuk server game Anda?

Salah satu aspek paling unik dari beban kerja game, dibandingkan dengan jenis beban kerja lainnya, adalah server game. Server game sangat penting untuk pengalaman pemain, karena pemain terhubung dengannya dari klien game mereka untuk memainkan sesi permainan.

Server game juga merupakan salah satu pendorong biaya terbesar untuk mengoperasikan game multipemain. Oleh karena itu, penting untuk mengoptimalkan cara Anda memanfaatkan infrastruktur komputasi untuk server game Anda untuk mengurangi biaya.

Praktik terbaik

- GAMECOST02-BP01 Optimalkan biaya transfer data di internet
- GAMECOST02-BP02 Optimalkan jumlah sesi game yang dihosting di setiap instance server game untuk mengoptimalkan biaya

- GAMECOST02-BP03 Pilih opsi harga komputasi yang sesuai untuk mengurangi biaya

## GAMECOST02-BP01 Optimalkan biaya transfer data di internet

Meskipun AWS terutama mengenakan biaya untuk transfer data keluar (keluar) dari AWS sumber daya Anda ke internet, perusahaan game dapat menghadapi biaya tinggi terkait dengan transfer data melalui AWS Direct Connect atau penyeimbang beban AWS Gateway, yang mungkin mengenakan biaya untuk data masuk (masuk) dan keluar. Terapkan solusi yang mengurangi biaya keseluruhan untuk mentransfer data dari AWS backend game Anda ke pemain Anda, dengan fokus meminimalkan biaya keluar dari AWS sumber daya Anda serta mengevaluasi opsi untuk mengelola biaya masuk dan keluar melalui layanan konektivitas. AWS

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Gunakan Amazon CloudFront untuk mengurangi biaya pengiriman konten dan aplikasi web yang menghadap publik.

Konten game dan aset yang disimpan di cloud biasanya disimpan di Amazon S3 dan dikirim ke klien game baik langsung dari S3 atau dari server web yang dihosting di Amazon EC2 yang mengambil konten dari Amazon S3 dan mengirimkannya ke klien. Untuk mengurangi biaya transfer data unduhan konten, pertimbangkan untuk menggunakan Amazon CloudFront di depan penyimpanan cloud Anda untuk mengirimkan konten kepada pengguna.

Penggunaan CloudFront dapat mengurangi biaya transfer data karena biaya pengiriman konten Anda lebih murah CloudFront points-of-presence daripada langsung dari Wilayah, dan CloudFront tidak membebankan biaya pengambilan asal untuk asal AWS berbasis, seperti Amazon EC2 dan Amazon S3. Jika konten Anda statis dan tidak sering berubah, Anda dapat menggunakan CloudFront cache data yang lebih dekat ke pengguna akhir, yang selanjutnya dapat mengurangi biaya.

CloudFront juga meningkatkan efisiensi biaya aplikasi dan layanan web yang menghadap publik yang menghadap ke depan, bahkan jika caching tidak digunakan, karena biaya transfer data antara server dan klien Anda dapat dikurangi dengan merutekan lalu lintas melalui jaringan. AWS

Amazon CloudWatch dapat digunakan untuk memantau CloudFront penggunaan Amazon Anda. Untuk kasus penggunaan di mana Anda menggunakan beberapa jaringan pengiriman konten (CDNs), Amazon CloudFront Origin Shield dapat menyediakan lapisan caching tambahan untuk mengkonsolidasikan dan mengurangi jumlah permintaan asal dari penyedia yang berbeda.

Untuk memahami lalu lintas jaringan game Anda, Anda dapat mengaktifkan VPC Flow Logs dan Amazon CloudWatch Internet Monitor agar memiliki end-to-end visibilitas pada koneksi backend pemain atau game. Pendekatan itu dapat mengidentifikasi penyebab tingginya biaya transfer data dan melakukan perubahan arsitektur untuk mengoptimalkan pengeluaran transfer data.

Langkah-langkah implementasi

- Gunakan Amazon CloudFront di depan Amazon S3 atau asal konten EC2 berbasis untuk mengurangi biaya transfer data dengan memanfaatkan pengiriman berbiaya lebih rendah dari CloudFront points-of-presence dan menghapus biaya pengambilan asal.
- Aktifkan VPC Flow Logs dan Amazon CloudWatch Internet Monitor untuk menganalisis lalu lintas jaringan dan mengidentifikasi perubahan arsitektur untuk mengoptimalkan biaya transfer data.
- Terapkan CloudFront Origin Shield untuk mengkonsolidasikan dan mengurangi permintaan asal saat menggunakan beberapa CDNs untuk efisiensi biaya tambahan.

Untuk praktik terbaik lainnya untuk pengiriman konten, lihat whitepaper Pengiriman Konten untuk Game.

## GAMECOST02-BP02 Optimalkan jumlah sesi game yang dihosting di setiap instance server game untuk mengoptimalkan biaya

Optimalkan jumlah sesi game yang dihosting per instance server untuk mencapai pemanfaatan komputasi yang lebih baik dan mengurangi biaya infrastruktur komputasi.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

### Panduan implementasi

Untuk mengoptimalkan biaya, pengembang game harus memaksimalkan jumlah sesi game yang dihosting di server fisik atau virtual yang sama, juga dikenal sebagai kepadatan pengepakan server game mereka. Hal ini dicapai dengan meningkatkan jumlah proses server game yang dapat secara bersamaan di-host pada sebuah instance.

Proses server game tunggal biasanya tidak memerlukan penggunaan seluruh sumber daya yang tersedia pada EC2 instance. Ini adalah salah satu cara paling penting untuk mengurangi biaya komputasi untuk permainan dan memerlukan penggunaan perangkat lunak yang dapat menelurkan dan mengelola beberapa proses server pada EC2 Instans pada port terpisah.

Misalnya, Amazon GameLift memiliki kuota pada jumlah maksimum proses server game per instance, yang harus Anda coba manfaatkan sehingga Anda dapat mengurangi biaya hosting. Untuk informasi selengkapnya, lihat titik akhir dan kuota GameLift Server Amazon untuk detail tentang kuota saat ini untuk proses server game maksimum per instans.

Sebagai alternatif untuk menyebarkan proses server game pada mesin virtual seperti EC2 instance, menjadi populer bagi pengembang game untuk menjalankan server game mereka sebagai aplikasi berbasis kontainer menggunakan solusi orkestrasi kontainer. Pengembang game dapat menggunakan Amazon Elastic Container Service (Amazon ECS) atau Panduan untuk Hosting Server Game Menggunakan Agones dan Open Match di Amazon EKS. Pilihan lainnya adalah Game Server Hosting on AWS Fargate, mesin komputasi tanpa server yang bekerja dengan ECS dan EKS, memungkinkan Anda untuk fokus pada game Anda tanpa harus mengelola infrastruktur yang mendasarinya.

Solusi kontainer menyediakan fungsionalitas penjadwalan pekerjaan yang dapat secara otomatis menemukan instance kontainer yang tersedia di cluster untuk meng-host wadah server game Anda berdasarkan persyaratan sumber daya dan logika penempatan lain yang Anda tentukan. Namun, penting untuk mempertimbangkan bagaimana Anda akan mengelola perilaku penskalaan dan penempatan pemain dengan cara yang tidak mengganggu sesi pemain aktif.

Langkah-langkah implementasi

- Tingkatkan kepadatan pengepakan dengan menjalankan beberapa proses server game per EC2 instance menggunakan port terpisah dan perangkat lunak manajemen proses.
- Gunakan Amazon GameLift atau solusi kontainer seperti ECS, EKS, atau AWS Fargate untuk mengelola proses server game secara efisien dan mengurangi biaya infrastruktur.
- Terus memantau pemanfaatan sumber daya untuk menyempurnakan kepadatan pengepakan dan mempertahankan efisiensi biaya tanpa mengorbankan pengalaman pemain.

## GAMECOST02-BP03 Pilih opsi harga komputasi yang sesuai untuk mengurangi biaya

Jalankan tes kinerja perangkat lunak server game Anda di berbagai jenis instans dan opsi komputasi untuk menentukan opsi mana yang paling hemat biaya untuk game Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Selain memanfaatkan jenis EC2 instans yang tepat secara efisien untuk beban kerja Anda, pertimbangkan opsi harga komputasi mana yang paling cocok untuk tujuan pengoptimalan biaya Anda. Ada beberapa opsi harga yang tersedia, termasuk Instans Sesuai Permintaan, Instans Spot, Instans Cadangan, dan Savings Plans.

Savings Plans (SPs) memberikan diskon untuk komputasi dengan membuat komitmen penggunaan dan ideal untuk skenario ketika Anda tidak dapat memperkirakan penggunaan yang diharapkan untuk periode 1 tahun atau 3 tahun. Mereka memberikan diskon seperti Instans Cadangan dengan fleksibilitas untuk menerapkan diskon ini di seluruh Wilayah, misalnya keluarga, sistem operasi, sewa. Mereka juga dapat diterapkan pada AWS Fargate, siapa yang bisa menjadi opsi hosting server game untuk game kasual atau AWS Lambda yang digunakan sebagai opsi bagus untuk game berbasis giliran yang tidak memerlukan server game. Untuk informasi selengkapnya, lihat Membangun game multi-pemain tanpa server yang berskala.

Savings Plans diperkenalkan selama peluncuran game untuk menghemat biaya beban kerja server game yang berkontribusi pada pengeluaran EC2 instance saat game dirilis ke audiens. Savings Plans juga dapat diperkenalkan pasca-peluncuran ketika tim operasi game memiliki pemahaman yang lebih baik tentang lalu lintas pemain setelah game telah berjalan dalam produksi untuk waktu yang lama.

Karena Savings Plans memberikan fleksibilitas regional, mereka sangat ideal untuk mengoptimalkan pengeluaran server game untuk game dengan penggunaan yang tidak terduga di seluruh wilayah.

Misalnya, jika pola penggunaan pemain harian Anda membutuhkan setidaknya 20 server untuk mendukung basis pemain Anda, tetapi secara berkala membutuhkan hingga 40 server, maka pertimbangkan untuk membeli komitmen Savings Plan untuk menutupi dasar 20 server, karena permintaan penggunaan itu dapat diprediksi dan konsisten, dan akan menghasilkan pemanfaatan maksimum komitmen penggunaan yang telah Anda beli.

Maksimalkan pemanfaatan Savings Plans dan tambahkan dengan opsi pembelian lain yang memberikan fleksibilitas lebih untuk lonjakan penggunaan server game yang tidak terduga, seperti on-demand dan Spot Instances untuk mencapai penghematan optimal.

Instans Spot ideal untuk menjalankan server game karena mereka menawarkan diskon komputasi terbesar, tidak memerlukan komitmen penggunaan, dan mereka memberikan fleksibilitas untuk jenis beban kerja yang tidak terduga dan runcing. Namun, Instans Spot dapat terganggu, sehingga paling cocok untuk beban kerja server game dengan durasi sesi permainan pendek atau situasi di mana toleransi untuk interupsi lebih tinggi.

Untuk informasi selengkapnya tentang panduan menjalankan server game menggunakan Kubernetes di Amazon EKS dengan Instans EC2 Spot, lihat Cara menjalankan game multipemain masif dengan Spot menggunakan Aurora Tanpa Server. EC2

Gunakan Instans EC2 Spot Amazon untuk menentukan kumpulan dengan kemungkinan gangguan paling sedikit yang akan memberikan penghematan maksimum dibandingkan dengan tarif sesuai permintaan.

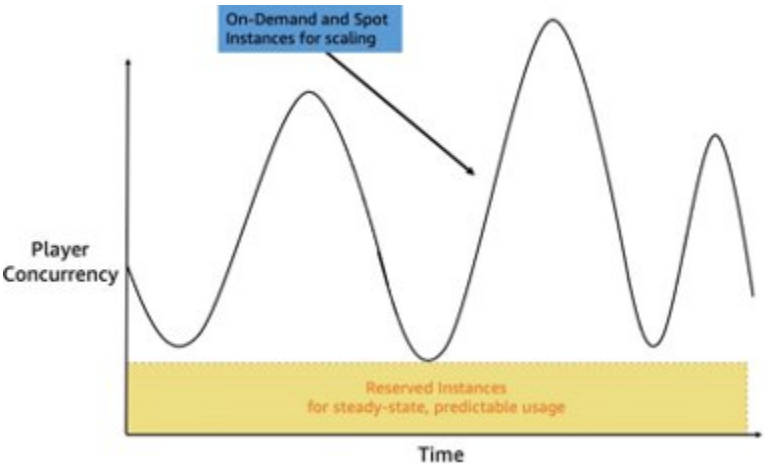
Saat menggunakan Spot, Anda juga disarankan untuk menjalankan beban kerja server game di beberapa jenis EC2 instans dan Availability Zone Wilayah AWS untuk mendiversifikasi penggunaan kapasitas Anda dan mengurangi risiko interupsi.

Pertimbangkan untuk menggunakan Instans Spot dalam kombinasi dengan Instans Sesuai Permintaan untuk meminimalkan dampak potensi gangguan pada sesi permainan aktif dan menggunakan strategi alokasi kapasitas yang dioptimalkan untuk mengurangi risiko gangguan lebih lanjut.

Lihat Praktik terbaik untuk Amazon EC2 Spot untuk praktik terbaik tambahan. Penyeimbangan Kembali Kapasitas dalam Auto Scaling untuk mengganti Instans Spot yang berisiko dapat digunakan untuk memantau secara proaktif dan menambah kapasitas tambahan saat Instans Spot berisiko tinggi mengalami gangguan.

Amazon GameLift FleetiQ terintegrasi dengan Instans Spot untuk mengoptimalkan penggunaan Instans Spot berbiaya rendah sekaligus mengurangi risiko interupsi. Jika Anda menghosting game Anda menggunakan GameLift, tinjau GameLift dokumentasi untuk memilih sumber daya komputasi. Untuk informasi selengkapnya, lihat Memilih sumber daya komputasi untuk armada terkelola.

Diagram berikut memberikan contoh untuk menggambarkan penggunaan beberapa opsi harga komputasi untuk beban kerja server game:



Hosting server game dengan beberapa opsi EC2 harga

Dalam diagram, konkurensi pemain berfluktuasi dari waktu ke waktu yang membuatnya sulit untuk mengelola pemanfaatan dan mencapai optimalisasi biaya. Untuk mengatasi fluktuasi ini, pertimbangkan untuk mengadopsi campuran opsi harga komputasi yang berbeda, menggunakan Savings Plans EC2 untuk memenuhi kebutuhan persyaratan penggunaan minimum Anda sambil mengandalkan Instans EC2 On-Demand dan EC2 Spot untuk memenuhi kebutuhan permintaan pemain Anda.

Langkah-langkah implementasi

- Gunakan Savings Plans untuk penggunaan dasar yang dapat diprediksi, gabungkan dengan Instans Spot dan Sesuai Permintaan untuk fleksibilitas dan optimalisasi biaya selama lonjakan penggunaan.
- Gunakan Instans Spot untuk server game dengan durasi sesi pendek atau toleransi interupsi yang lebih tinggi, diversifikasi di seluruh jenis instans dan Availability Zone untuk meminimalkan risiko.
- Terapkan alat seperti EC2 Spot Instances Advisor, Capacity Rebalancing, dan GameLift FleetiQ untuk mengoptimalkan penggunaan Instans Spot dan mengelola interupsi secara proaktif.

## Biaya transfer data

GAMECOST03: Bagaimana Anda mengoptimalkan biaya transfer data untuk infrastruktur game Anda?

Game dapat mentransfer sejumlah besar data di internet antara perangkat klien game pemain Anda dan infrastruktur game Anda untuk memberikan pengalaman bermain game, serta di antara komponen infrastruktur game Anda.

Misalnya, transfer data terjadi ketika pemain mengunduh pembaruan konten game ke klien game mereka, menyimpan status kemajuan game mereka ke cloud, terlibat dalam sesi permainan multipemain waktu nyata dengan teman-teman mereka, dan ketika infrastruktur game Anda mentransfer data antara Wilayah dan Zona Ketersediaan. Penting untuk memahami di mana transfer data terjadi dalam beban kerja game Anda untuk mengoptimalkan pilihan arsitektur Anda untuk mengurangi biaya transfer data ini.

Untuk mengoptimalkan biaya transfer data untuk beban kerja game Anda, pertimbangkan praktik terbaik berikut:

Praktik terbaik

- GAMECOST03-BP01 Pilih jenis penyimpanan yang sesuai untuk konten buatan pengguna untuk mengurangi biaya
- GAMECOST03-BP02 Optimalkan database untuk backend game

## GAMECOST03-BP01 Pilih jenis penyimpanan yang sesuai untuk konten buatan pengguna untuk mengurangi biaya

Setiap jenis data yang dihasilkan dan disimpan dalam game Anda memiliki karakteristik unik yang harus Anda pertimbangkan saat menentukan solusi penyimpanan yang tepat untuk beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Rendah

### Panduan implementasi

Gunakan Amazon S3 Object Lifecycle Management untuk menyimpan data objek di kelas penyimpanan yang paling hemat biaya. Amazon S3 menyediakan beberapa kelas penyimpanan dan manajemen siklus hidup objek untuk memudahkan pengaturan kebijakan sederhana dan halus untuk secara otomatis mentransisikan data antar tingkatan penyimpanan untuk mengurangi biaya. Alih-alih hanya menyimpan data di kelas penyimpanan standar S3 secara default, pertimbangkan untuk menyiapkan konfigurasi siklus hidup untuk mentransisikan data antar tingkatan secara otomatis dari waktu ke waktu, atau gunakan kelas penyimpanan S3 Intelligent-Tiering untuk pola akses yang tidak diketahui atau berubah.

Atau, S3 Intelligent-Tiering dapat secara hemat biaya dan secara otomatis mentransisikan data antar tingkatan dan direkomendasikan sebagai kelas penyimpanan default karena menyediakan pengoptimalan biaya tanpa perlu mengatur kebijakan siklus hidup secara manual, dan sekarang menjadi pilihan terbaik untuk objek kecil dan berumur pendek. Untuk informasi selengkapnya, lihat Amazon S3 Intelligent-Tiering — Peningkatan Pengoptimalan Biaya untuk Objek Berumur Pendek dan Kecil.

Kasus penggunaan umum untuk Amazon S3 termasuk penyimpanan aset game, konten statis, log game, penyimpanan data lake, dan cadangan. Untuk kasus penggunaan di mana sistem

file diperlukan, seperti melampirkan sistem file bersama ke workstation selama pengembangan, pertimbangkan untuk menggunakan Amazon Elastic File System (Amazon EFS), yang menyediakan kelas penyimpanan yang berbeda dan secara otomatis tumbuh dan menyusut saat Anda menambahkan dan menghapus file tanpa perlu mengelola infrastruktur.

Amazon S3 One Zone -IA adalah opsi penyimpanan ideal untuk data sementara yang terkait dengan sesi dalam game, perjodohan, atau informasi singkat lainnya yang dapat dibuat ulang sesuai kebutuhan. Jenis data game tersebut tidak memerlukan redundansi di beberapa Availability Zones ()AZs. Kelas penyimpanan berbiaya lebih rendah ini sangat cocok untuk catatan aksi pemain, peristiwa game, dan data telemetri lainnya yang digunakan untuk analitik atau debugging.

Manfaat optimalisasi biaya utama menggunakan S3 Express One Zone untuk data game tersebut adalah penghematan biaya yang signifikan dibandingkan dengan kelas penyimpanan S3 standar, dengan pengurangan biaya penyimpanan hingga 20%. Ini bisa sangat menguntungkan untuk game dengan volume data besar yang tidak memerlukan tingkat daya tahan dan ketersediaan yang sama dengan data aplikasi mission-critical. Dengan memanfaatkan S3 One Zone, pengembang dan penerbit game dapat mengoptimalkan biaya penyimpanan cloud mereka tanpa mengorbankan pengalaman pemain secara keseluruhan.

Langkah-langkah implementasi

- Konfigurasikan kebijakan siklus hidup Amazon S3 untuk mentransisikan data antar kelas penyimpanan atau gunakan S3 Intelligent-Tiering sebagai default untuk pengoptimalan biaya otomatis dengan mengubah pola akses.
- Gunakan S3 One Zone-Infrequent Access untuk data sesi permainan sementara, seperti telemetri dan catatan perjodohan, untuk mengurangi biaya penyimpanan hingga 20% sambil mempertahankan ketersediaan yang memadai.
- Untuk kebutuhan sistem file bersama selama pengembangan, gunakan Amazon EFS untuk menyederhanakan manajemen penyimpanan dengan kapasitas elastis dan beberapa kelas penyimpanan.

## GAMECOST03-BP02 Optimalkan database untuk backend game

Game sangat bergantung pada database untuk menyimpan berbagai data penting, mulai dari profil pemain dan inventaris hingga transaksi mikro dalam game dan metrik perkembangan. Database juga memainkan peran penting dalam mengelola aspek sosial permainan, seperti membuat dan memelihara kelompok pemain, partai, dan menegakkan kebijakan moderasi. Ketika basis pemain

permainan tumbuh, biaya database terkait pasti akan meningkat untuk mengakomodasi peningkatan permintaan data dan penggunaan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Sedang

## Panduan implementasi

Untuk backend game yang berjalan di Amazon Aurora, ada beberapa strategi pengoptimalan biaya yang dapat digunakan. Salah satu rekomendasi utama adalah untuk menskalakan replika baca Anda secara otomatis berdasarkan pola penggunaan, secara dinamis menskalakan jumlah replika ke atas atau ke bawah untuk menangani fluktuasi lalu lintas. Ini berarti Anda membayar untuk sumber daya yang benar-benar Anda butuhkan. Taktik pengoptimalan lainnya adalah mengganti replika baca yang digunakan untuk analitik game dengan ekspor snapshot DB ke Amazon S3, karena layanan penyimpanan S3 umumnya lebih terjangkau daripada instance database Aurora yang disediakan. Untuk informasi selengkapnya, lihat Mengekspor data snapshot DB ke Amazon S3 untuk Amazon RDS.

Menjelajahi penggunaan instans DB Cadangan untuk Amazon Aurora untuk instans basis data inti Anda dan beralih ke konfigurasi Tanpa Server Aurora juga dapat menghasilkan penghematan biaya jangka panjang yang besar dengan memberikan lebih banyak fleksibilitas dan kontrol terperinci atas pemanfaatan sumber daya Anda.

Demikian pula, untuk backend game yang menggunakan Amazon DynamoDB, menggunakan mode kapasitas sesuai permintaan DynamoDB dapat menjadi pilihan yang efektif, terutama untuk beban kerja baru atau tidak dapat diprediksi, karena memungkinkan Anda membayar hanya untuk sumber daya yang Anda konsumsi tanpa perlu penyediaan berlebihan. Karena pola lalu lintas game Anda menjadi lebih stabil dan dapat diprediksi dari waktu ke waktu, Anda kemudian dapat beralih ke mode kapasitas yang disediakan DynamoDB, yang dapat menawarkan penghematan biaya melalui perencanaan kapasitas yang lebih baik. Mengaktifkan auto-scaling pada tabel DynamoDB Anda adalah pengoptimalan kunci lainnya, memungkinkan layanan untuk secara dinamis menyesuaikan kapasitas yang disediakan berdasarkan fluktuasi lalu lintas. Uji struktur data game Anda di lingkungan pengembangan sebelum diluncurkan untuk menemukan dan menghapus indeks sekunder lokal (LSIs) dan indeks sekunder global () yang tidak perlu. GSIs Hal ini dapat menyebabkan penghematan biaya yang besar untuk penyimpanan dan operasi data game. Menghapus operasi Pindai yang tidak efisien dari kode backend game Anda demi kueri yang lebih bertarget, membeli kapasitas cadangan Amazon DynamoDB, dan memanfaatkan DynamoDB Streams dengan pemicu untuk memproses peristiwa backend game dapat lebih mengoptimalkan biaya DynamoDB Anda. AWS Lambda Untuk informasi selengkapnya, lihat Praktik terbaik untuk kueri dan pemindaian data di DynamoDB.

Dengan menerapkan strategi pengoptimalan biaya ini untuk Amazon Aurora dan DynamoDB, pengembang dan penerbit game dapat secara signifikan mengurangi pengeluaran basis data backend game mereka.

Langkah-langkah implementasi

- Gunakan auto-scaling replika baca Aurora dan ekspor snapshot DB ke Amazon S3 untuk penanganan lalu lintas yang berfluktuasi dan kebutuhan analitik yang hemat biaya.
- Optimalkan biaya DynamoDB dengan memulai dengan kapasitas sesuai permintaan untuk beban kerja baru, transisi ke kapasitas yang disediakan dengan auto-scaling untuk lalu lintas yang dapat diprediksi, dan menghapus yang tidak terpakai dan. LSIs GSIs
- Hindari operasi Pemindaian yang tidak efisien demi kueri yang ditargetkan, gunakan Instans Cadangan atau Kapasitas Cadangan, dan gunakan DynamoDB Streams untuk pemrosesan acara. AWS Lambda

## Mengelola permintaan dan sumber daya pasokan

Tidak ada pengelolaan permintaan dan sumber daya pasokan praktik terbaik khusus untuk Game Lens.

Untuk informasi selengkapnya tentang mengelola permintaan dan penyediaan sumber daya, lihat Pilar Pengoptimalan Biaya - Kerangka Kerja AWS Well-Architected.

## Mengoptimalkan dari waktu ke waktu

Tidak ada optimasi dari waktu ke waktu praktik terbaik khusus untuk Game Lens.

Untuk informasi selengkapnya tentang mengoptimalkan biaya dari waktu ke waktu, lihat Pilar Pengoptimalan Biaya - Kerangka Kerja yang AWS Dirancang dengan Baik.

## Sumber daya

Lihat sumber daya berikut untuk mempelajari lebih lanjut tentang praktik terbaik untuk pengoptimalan biaya:

Dokumen terkait:

- Bagaimana cara mengurangi biaya transfer data untuk gateway NAT saya di Amazon VPC?

- Memperkenalkan adaptor Amazon GameLift FleetiQ untuk Agones
- Bagaimana cara menemukan kontributor teratas untuk lalu lintas gateway NAT di VPC Amazon saya?
- Pilih strategi komputasi yang tepat untuk server game global Anda
- AWS Well-Architected Labs — Sumber daya hemat biaya
- Plugin Amazon VPC CNI meningkatkan batas pod per node
- Praktik Terbaik Arsitektur untuk Optimalisasi Biaya
- Mengurangi waktu tunggu pemain dan alokasi komputasi ukuran yang tepat menggunakan Amazon SageMaker AI RL dan Amazon EKS
- AWS Compute Optimizer
- Electronic Arts mengoptimalkan biaya penyimpanan dan operasi menggunakan Amazon S3 Intelligent-Tiering dan Amazon Glacier
- Melarikan diri dari praktik lisensi yang tidak ramah dengan memigrasikan beban kerja Windows ke Linux
- Ikhtisar Biaya Transfer Data untuk Arsitektur Umum
- AWS dan Kubecost berkolaborasi untuk memberikan pemantauan biaya bagi pelanggan EKS
- Meletakkan Yayasan: Menyiapkan Lingkungan Anda untuk Optimalisasi Biaya
- Ikhtisar Instans EC2 Spot Amazon

# Keberlanjutan

Pilar keberlanjutan menyediakan prinsip-prinsip desain, panduan operasional, praktik terbaik, dan rencana peningkatan untuk membantu memenuhi target keberlanjutan beban kerja Anda. AWS

Anda dapat menemukan panduan implementasi tentang implementasi di whitepaper Sustainability Pillar - AWS Well-Architected Framework.

Area fokus

- Prinsip desain
- Pemilihan wilayah
- Penyelarasan dengan permintaan
- Perangkat lunak dan arsitektur
- Manajemen data
- Perangkat keras dan layanan
- Sumber daya

## Prinsip desain

Keberlanjutan untuk industri game berkembang pada tingkat yang beragam di berbagai Wilayah di dunia. Dengan kekuatan berkelanjutan di petak besar Amerika Utara dan mandat keberlanjutan di UE dan Inggris, arsitek memiliki beberapa pendekatan untuk mencapai beban kerja yang lebih hijau dalam waktu dekat. Pilar keberlanjutan Well-Architected dapat digunakan untuk mencapai langkah-langkah ini untuk berbagai macam beban kerja.

Bagian lensa ini menjelaskan beberapa praktik terbaik yang dapat digunakan untuk beban kerja game.

- Pilih jenis penyimpanan yang sesuai untuk data pengguna game.
- Waspadai kebijakan siklus hidup data yang akan menghapus duplikat data dan membersihkan data yang tidak dibutuhkan dari beban kerja Anda.
- Selektif tentang ukuran yang tepat penyebaran sumber daya komputasi.
- Gunakan tanpa server untuk proses singkat dan transaksional.

## Pemilihan wilayah

Tidak ada praktik terbaik pemilihan Wilayah khusus untuk Game Lens. Untuk detail selengkapnya, lihat Pilar Keberlanjutan - Kerangka AWS Well-Architected.

## Penyelarasan dengan permintaan

Tidak ada keselarasan untuk menuntut praktik terbaik khusus untuk Game Lens. Untuk detail selengkapnya, lihat Pilar Keberlanjutan - Kerangka AWS Well-Architected.

## Perangkat lunak dan arsitektur

Tidak ada praktik terbaik perangkat lunak dan arsitektur khusus untuk Game Lens. Untuk detail selengkapnya, lihat Pilar Keberlanjutan - Kerangka AWS Well-Architected.

## Manajemen data

GAMESUS01: Bagaimana Anda mengelola data pengguna dan game di sistem game Anda?

Mengembangkan strategi siklus hidup data yang mengoptimalkan penyimpanan data dan relevansi dengan hanya mempertahankan data historis penting.

Praktik terbaik

- GAMESUS01-BP01 Gunakan teknologi penyimpanan yang sesuai dengan pola yang disesuaikan dengan konten pengguna, informasi pelanggan, dan pembelian dalam game
- GAMESUS01-BP02 Gunakan kebijakan siklus hidup atau kedaluwarsa TTL untuk menghapus data pengguna game yang tidak perlu, file log, atau aset usang

### GAMESUS01-BP01 Gunakan teknologi penyimpanan yang sesuai dengan pola yang disesuaikan dengan konten pengguna, informasi pelanggan, dan pembelian dalam game

Anda harus mengklasifikasikan data Anda berdasarkan jenis, kebutuhan retensi, dan frekuensi akses. Ini memungkinkan Anda untuk memilih solusi penyimpanan yang paling dioptimalkan untuk

berbagai jenis data dari game atau produk layanan backend Anda. Data yang berubah cepat harus disimpan dalam layanan basis data nilai kunci atau dalam memori. Data transaksional harus disimpan dalam layanan database relasional. File besar, aset game, atau konten buatan pengguna harus disimpan dalam layanan penyimpanan objek.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Game memproduksi dan mengkonsumsi berbagai macam tipe data yang memerlukan solusi penyimpanan yang dioptimalkan untuk frekuensi akses, latensi, dan biaya. Data yang disimpan harus diklasifikasikan menggunakan tag untuk membedakan data yang dapat dihapus atau perlu disimpan dalam jangka panjang.

Layanan berikut bekerja dengan baik untuk berbagai kasus penggunaan Game:

Amazon Aurora (kompatibel dengan MySQL dan PostgreSQL) menawarkan ketersediaan tinggi, latensi rendah, dan penskalaan otomatis, menjadikannya pilihan yang sangat baik untuk menangani sejumlah besar data transaksional, seperti manajemen dan otentikasi akun pemain, ekonomi dalam game, papan peringkat dan peringkat pemain, ketekunan status game, manajemen acara dan kampanye, serta penyebaran multi-wilayah dan ketersediaan tinggi.

Amazon DynamoDB adalah database NoSQL yang dikelola sepenuhnya yang dikenal dengan latensi rendah, throughput tinggi, dan skalabilitas yang mulus, yang membuatnya ideal untuk menangani data pemain waktu nyata, manajemen sesi, inventaris, ekonomi dalam game, status game multipemain waktu nyata, perjodohan, pencatatan peristiwa, dan penskalaan untuk pemirsa global.

Amazon DocumentDB (kompatibel dengan MongoDB) menyediakan layanan database berorientasi dokumen latensi rendah yang skalabel, sempurna untuk menyimpan data semi-terstruktur yang fleksibel, seperti sistem inventaris, profil pemain dan kustomisasi, dunia game dan konten yang dihasilkan secara prosedural, interaksi sosial dan pemain, analitik dan pelacakan perilaku, serta metadata dan konfigurasi dalam game.

Amazon ElastiCache mendukung caching dalam memori dengan Redis atau Memcached, menawarkan akses data yang cepat dan waktu respons yang berkurang, yang sangat penting untuk game multipemain waktu nyata di mana kecepatan dan kinerja sangat penting untuk pengalaman pengguna yang lancar. ElastiCache digunakan dalam game untuk papan peringkat real-time, manajemen sesi, metadata game caching, obrolan dan pesan dalam game, perjodohan, analitik dan telemetri waktu nyata, dan penskalaan untuk acara dengan lalu lintas tinggi.

Amazon Simple Storage Service (S3) dapat digunakan untuk menyimpan objek seperti aset game, video, gambar, file log teks, dan lainnya. S3 adalah layanan penyimpanan objek yang menawarkan skalabilitas, ketersediaan data, keamanan, dan kinerja terdepan di industri.

Jika menawarkan beberapa kelas penyimpanan yang mendukung akses data yang sering dan sering, dan penyimpanan arsip yang hemat biaya. Untuk data yang sering diakses selama pengembangan, studio harus menyimpan objek dalam Standar S3 untuk latensi rendah dan kinerja throughput tinggi. Untuk data yang sering berubah dari panas ke dingin atau sebaliknya, studio harus menyelidiki S3 Intelligent-Tiering. Intelligent-Tiering memantau pola akses data Anda dan secara otomatis memindahkan data ke tingkat akses yang paling hemat biaya.

Untuk studio yang membutuhkan throughput tinggi, latensi rendah, dan tidak masalah jika tinggal di satu Availability Zone, gunakan S3 Express One Zone. Ini mereplikasi data ke satu AZ dan dapat meningkatkan kecepatan akses data dibandingkan dengan standar S3. Untuk kebutuhan arsip mendalam data historis Amazon juga menawarkan Amazon Glacier. Kelas penyimpanan Amazon Glacier dibuat khusus untuk pengarsipan data, memberi Anda kinerja tinggi, fleksibilitas pengambilan, dan penyimpanan arsip berbiaya rendah di cloud.

Amazon Elastic Block Store dapat digunakan untuk menyimpan binari server game, file yang dapat dieksekusi, dan konfigurasi server game atau repositori aset Anda perlu berfungsi. Anda harus memotret dan menghapus volume yang tidak digunakan yang tidak dilampirkan ke instance EC2 . Ini membebaskan Anda dari biaya penyimpanan yang dikeluarkan sambil mengurangi penggunaan layanan dan perangkat keras yang tidak dibutuhkan.

Langkah-langkah implementasi

- Mengklasifikasikan data game berdasarkan jenis, kebutuhan retensi, dan frekuensi akses, menandai data untuk membedakan antara persyaratan penyimpanan jangka pendek dan jangka panjang.
- Gunakan Amazon Aurora untuk data transaksional, DynamoDB untuk data pemutar real-time, DocumentDB untuk data semi-terstruktur, dan untuk cache latensi rendah informasi game yang kritis waktu. ElastiCache
- Simpan aset game, log, dan konten buatan pengguna di Amazon S3, pilih kelas penyimpanan yang sesuai (misalnya, Intelligent-Tiering, One Zone, dan Glacier) berdasarkan pola akses dan kebutuhan arsip, dan gunakan EBS untuk binari dan konfigurasi server game dengan manajemen snapshot reguler.

# GAMESUS01-BP02 Gunakan kebijakan siklus hidup atau kedaluwarsa TTL untuk menghapus data pengguna game yang tidak perlu, file log, atau aset usang

Anda dapat menggunakan tag dan tipe data untuk membuat kebijakan siklus hidup atau TTL untuk memindahkan data ke penyimpanan arsip atau menghapus sepenuhnya dari layanan. Ini mungkin termasuk konfigurasi sementara, konten arsip kedaluwarsa, dan log historis yang tidak lagi diperlukan. Sebagian besar layanan mendukung penandaan.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

## Panduan implementasi

Untuk data yang disimpan di S3, Anda dapat menggunakan kebijakan siklus hidup untuk memindahkan data ke tingkat penyimpanan akses dan arsip yang jarang. Dalam konfigurasi Siklus Hidup S3, Anda dapat menentukan aturan untuk transisi objek dari satu kelas penyimpanan ke kelas penyimpanan lainnya guna menghemat biaya penyimpanan. Saat Anda tidak mengetahui pola akses objek Anda, atau pola akses Anda berubah seiring waktu berjalan, Anda dapat melakukan transisi objek ke kelas penyimpanan S3 Intelligent-Tiering untuk penghematan biaya otomatis.

Amazon S3 mendukung model air terjun untuk transisi antar kelas penyimpanan, seperti yang ditunjukkan pada diagram berikut.

Anda dapat menambahkan tindakan transisi ke konfigurasi Siklus Hidup S3 untuk memberi tahu Amazon S3 agar menghapus objek di akhir masa pakainya. Saat objek mencapai akhir masa pakainya berdasarkan konfigurasi siklus hidupnya, Amazon S3 mengambil tindakan Kedaluwarsa berdasarkan status Pembuatan Versi S3 tempat bucket berada:

- Bucket non-versioned: Amazon S3 mengantri objek untuk dihapus dan menghapusnya secara asinkron, menghapus objek secara permanen.
- Bucket berkemampuan versi: Jika versi objek saat ini bukan penanda hapus, Amazon S3 menambahkan penanda hapus dengan ID versi unik. Ini membuat versi saat ini menjadi versi lama, dan penanda hapus menjadi versi saat ini.
- Bucket yang ditangguhkan versi: Amazon S3 membuat penanda hapus dengan null sebagai ID versi. Penanda hapus ini menggantikan versi objek dengan ID versi null dalam hierarki versi, yang secara efektif menghapus objek.
- Ketika Anda menambahkan konfigurasi Siklus Hidup ke dalam bucket, aturan konfigurasi berlaku untuk objek yang ada dan objek yang Anda tambahkan kemudian. Misalnya, jika Anda

menambahkan aturan konfigurasi Siklus Hidup hari ini dengan tindakan kedaluwarsa yang menyebabkan objek dengan awalan tertentu kedaluwarsa 30 hari setelah pembuatan, Amazon S3 akan mengantri untuk menghapus objek yang ada yang berusia lebih dari 30 hari dan yang memiliki awalan yang ditentukan.

Time To Live (TTL) untuk DynamoDB adalah metode hemat biaya untuk menghapus item yang tidak lagi relevan. TTL memungkinkan Anda untuk menentukan stempel waktu kedaluwarsa per item yang menunjukkan kapan item tidak lagi diperlukan. DynamoDB secara otomatis menghapus item yang kedaluwarsa dalam beberapa hari dari waktu kedaluwarsa, tanpa menghabiskan throughput penulisan.

- Untuk menggunakan TTL, pertama-tama aktifkan pada tabel dan kemudian tentukan atribut tertentu untuk menyimpan stempel waktu kedaluwarsa TTL. Stempel waktu harus disimpan dalam format waktu epoch Unix pada perincian detik. Setiap kali item dibuat atau diperbarui, Anda dapat menghitung waktu kedaluwarsa dan menyimpannya di atribut TTL.
- Item dengan atribut TTL yang valid dan kedaluwarsa dapat dihapus oleh sistem, biasanya dalam beberapa hari setelah kedaluwarsa. Anda masih dapat memperbarui item kedaluwarsa yang menunggu penghapusan, termasuk mengubah atau menghapus atribut TTL mereka. Saat memperbarui item yang kedaluwarsa, kami menyarankan Anda menggunakan ekspresi kondisi untuk memastikan item tersebut belum dihapus selanjutnya. Gunakan ekspresi filter untuk menghapus item kedaluwarsa dari hasil Pindai dan Kueri.
- Item yang dihapus bekerja sama dengan yang dihapus melalui operasi penghapusan tipikal. Setelah dihapus, item masuk ke DynamoDB Streams sebagai penghapusan layanan alih-alih menghapus pengguna dan dihapus dari indeks sekunder lokal dan indeks sekunder global seperti operasi penghapusan lainnya.

Dengan ElastiCache untuk Redis Anda dapat mengontrol kesegaran data cache Anda dengan menggunakan TTLs atau kedaluwarsa pada kunci cache. Setelah waktu yang ditentukan berlalu, kunci dihapus dari cache, dan akses ke penyimpanan data asal diperlukan bersamaan dengan mencapai data yang diperbarui.

- Dua prinsip menentukan yang tepat TTLs untuk diterapkan dan jenis pola caching untuk diterapkan. Pertama, penting bagi Anda untuk memahami tingkat perubahan data yang mendasarinya. Kedua, penting bagi Anda untuk mengevaluasi risiko data usang dikembalikan ke aplikasi Anda alih-alih rekannya yang diperbarui.

- Dengan data dinamis yang sering berubah, Anda mungkin ingin menerapkan data TTLs yang kedaluwarsa yang lebih rendah pada tingkat perubahan yang cocok dengan database utama. Ini menurunkan risiko pengembalian data yang sudah ketinggalan zaman sambil tetap menyediakan buffer untuk membongkar permintaan database.
- Penting juga untuk menyadari bahwa, bahkan jika Anda hanya menyimpan data selama beberapa menit atau detik versus durasi yang lebih lama, menerapkan dengan tepat TTLs ke kunci cache Anda dapat menghasilkan peningkatan kinerja dan pengalaman pemain yang lebih baik secara keseluruhan dengan permainan Anda.

Langkah-langkah implementasi

- Gunakan kebijakan Siklus Hidup Amazon S3 untuk mentransisikan objek ke tingkat akses atau arsip yang jarang terjadi dan mengonfigurasi tindakan kedaluwarsa untuk menghapus objek yang tidak perlu berdasarkan aturan siklus hidup.
- Aktifkan Time to Live (TTL) di tabel DynamoDB untuk secara otomatis menghapus item kedaluwarsa tanpa menggunakan throughput tulis, menentukan stempel waktu kedaluwarsa dalam waktu epoch Unix.
- Tetapkan sesuai TTLs untuk ElastiCache kunci berdasarkan tingkat perubahan data dan toleransi risiko untuk data yang sudah ketinggalan zaman, memfasilitasi kesegaran data yang di-cache dan pengalaman pemain yang lebih baik.

## Perangkat keras dan layanan

GAMESUS02: Bagaimana Anda mengelola penggunaan sumber daya komputasi di backend game Anda?

Studio harus mengembangkan strategi komputasi yang menggunakan campuran berbagai jenis komputasi, layanan terkelola, dan rencana tabungan untuk mengoptimalkan penggunaan Anda. Anda juga harus mengoptimalkan bagaimana server game dan layanan backend dikemas untuk menghitung instance untuk mengurangi jumlah sumber daya yang tidak dibutuhkan.

Praktik terbaik

- GAMESUS02-BP01 Pilih layanan terkelola untuk beban kerja komputasi yang sesuai

- GAMESUS02-BP02 Ukuran komputasi Anda dengan benar dan gunakan kinerja GPU hanya jika diperlukan

## GAMESUS02-BP01 Pilih layanan terkelola untuk beban kerja komputasi yang sesuai

Arsitek layanan backend game Anda untuk menggunakan layanan terkelola untuk beban kerja lalu lintas yang didorong oleh peristiwa atau sangat bervariasi. Layanan terkelola menggeser manajemen infrastruktur ke AWS dan mendistribusikan dampak lingkungan di beberapa pengguna karena pesawat kontrol multi-penyewa.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

AWS Layanan seperti AWS Lambda, AWS Fargate (Container), dan Amazon Gamelift (Orkestrasi server game) dapat menjalankan kode, container, atau mengatur server game Anda tanpa harus mengelola infrastruktur yang mendasarinya. Layanan ini secara otomatis menskalakan berdasarkan permintaan pemain dan Anda hanya dikenakan biaya untuk sumber daya yang Anda konsumsi. Karena infrastruktur yang mendasarinya dikelola atas nama Anda, Anda hanya dapat fokus pada permainan dan persyaratan layanan backend Anda.

Anda dapat menggunakan AWS Lambdauntuk menjalankan kode tanpa menyediakan atau mengelola server. Lambda menjalankan kode Anda pada infrastruktur komputasi ketersediaan tinggi dan melakukan administrasi sumber daya komputasi, termasuk pemeliharaan server dan sistem operasi, penyediaan kapasitas dan penskalaan otomatis, dan logging. Dengan Lambda, Anda perlu menyediakan kode Anda di salah satu runtime bahasa yang didukung Lambda. Lambda berguna untuk memproses acara game, otentikasi pemain, pemrosesan pembelian dalam game, dan permintaan perjodohan. Lambda secara otomatis menskalakan berdasarkan jumlah peristiwa dan dapat menangani lonjakan lalu lintas yang tidak terduga.

AWS Fargateadalah mesin komputasi tanpa server untuk kontainer yang bekerja dengan Amazon Elastic Container Service (ECS) dan Amazon Elastic Kubernetes Service (EKS). AWS Fargate membuatnya mudah untuk fokus membangun aplikasi Anda dengan mengurangi kebutuhan untuk menyediakan dan mengelola server, memungkinkan Anda menentukan dan membayar sumber daya per aplikasi, dan meningkatkan keamanan melalui isolasi aplikasi berdasarkan desain. Fargate sangat ideal untuk layanan backend yang menangani profil pemain, manajemen negara, dan perjodohan.

Amazon GameLift adalah layanan terkelola untuk menyebarkan, mengoperasikan, dan menskalakan server game khusus untuk game multipemain berbasis sesi. Anda dapat menyebarkan server game pertama Anda di cloud hanya dalam beberapa menit, menghemat hingga ribuan jam teknik dalam pengembangan perangkat lunak di muka dan menurunkan risiko teknis yang sering menyebabkan pengembang memotong fitur multipemain dari desain mereka.

Langkah-langkah implementasi

- Gunakan AWS Lambda untuk beban kerja berbasis peristiwa seperti memproses acara game, otentikasi pemain, pembelian dalam game, dan permintaan perjodohan, memanfaatkan penskalaan otomatis dan manajemen tanpa server.
- Terapkan AWS Fargate dengan ECS atau EKS untuk layanan backend seperti profil pemain, manajemen negara, dan perjodohan, menghapus manajemen server dan meningkatkan isolasi aplikasi.
- Gunakan Amazon GameLift untuk menyebarkan dan menskalakan server game khusus untuk game multipemain berbasis sesi, mengurangi waktu pengembangan dan kompleksitas operasional.

## GAMESUS02-BP02 Ukuran komputasi Anda dengan benar dan gunakan kinerja GPU hanya jika diperlukan

Arsitek server game dan backend Anda untuk memanfaatkan sumber daya komputasi secara efisien. Komputasi penyediaan berlebih dapat menyebabkan biaya yang tidak perlu dan meminimalkan jumlah sumber daya yang tidak digunakan atau kurang dimanfaatkan. Instans GPU harus digunakan untuk mendukung upaya pengembangan khusus seperti membangun kembali HLOD di Unreal, atau jika server game Anda memerlukannya berdasarkan desain. Ini sangat mengurangi dampak lingkungan dan biaya beban kerja Anda.

Tingkat risiko yang terjadi jika praktik terbaik ini tidak diterapkan: Tinggi

### Panduan implementasi

Anda harus mengoptimalkan server game dan layanan backend Anda untuk menggunakan beberapa jenis EC2 instance dan jumlah instans paling sedikit yang diperlukan. Ini meningkatkan jumlah instance yang tersedia untuk memenuhi kebutuhan Anda selama pengembangan atau untuk peluncuran game Anda. Anda juga harus mencocokkan jenis instance dengan beban kerja tertentu yang Anda terapkan. Instans Compute Optimized mendukung berbagai kasus penggunaan termasuk server game dan layanan backend seperti perjodohan. Instans Memory Optimized dirancang untuk

memberikan kinerja cepat untuk beban kerja yang memproses kumpulan data besar dalam memori. Gunakan instans GPU seperti yang diperlukan untuk persyaratan kinerja tinggi, tetapi tidak untuk tugas komputasi umum. Jika memungkinkan, arsitek layanan atau server game Anda untuk berjalan di ARM dengan instance AWS Graviton. Graviton adalah jenis instans yang paling berkinerja hingga hemat energi yang tersedia di. AWS Mereka juga menawarkan peningkatan kinerja dan biaya jika dibandingkan dengan jenis instans x86.

Gunakan AWS Compute Optimizeruntuk mengidentifikasi konfigurasi AWS sumber daya yang optimal, seperti jenis instans Amazon Elastic Compute Cloud (EC2), konfigurasi volume Amazon Elastic Block Store (EBS), ukuran tugas AWS Fargate layanan Amazon Elastic Container Service (ECS), AWS Lambda lisensi perangkat lunak komersial, ukuran memori fungsi, dan kelas instans DB Amazon Relational Database Service (RDS), menggunakan machine learning menganalisis metrik pemanfaatan historis. Compute Optimizer menyediakan APIs serangkaian pengalaman konsol untuk mengurangi biaya dan meningkatkan kinerja beban kerja dengan merekomendasikan sumber daya optimal untuk beban kerja Anda. AWS AWS

Langkah-langkah implementasi

- Cocokkan sumber daya komputasi dengan beban kerja tertentu dengan menggunakan instans Compute Optimized untuk server game, instans Memory Optimized untuk kumpulan data besar, dan instance GPU hanya untuk tugas seperti rebuild HLOD atau server game yang bergantung pada GPU.
- Optimalkan pemanfaatan komputasi dengan menerapkan instans AWS Graviton jika memungkinkan untuk efisiensi energi, kinerja yang lebih baik, dan penghematan biaya dibandingkan dengan instans x86.
- Gunakan AWS Compute Optimizer untuk menganalisis pemanfaatan historis dan merekomendasikan konfigurasi yang paling efisien untuk beban kerja EC2, AWS ECS, AWS Lambda dan Amazon RDS untuk mengurangi biaya dan meningkatkan kinerja.

## Sumber daya

Lihat sumber daya berikut untuk mempelajari lebih lanjut tentang praktik terbaik kami terkait keberlanjutan.

- PBB: Industri game menyoroti ancaman terhadap planet ini
- Medium: Keberlanjutan Lingkungan dalam Pengembangan Game: Bermain Secara Bertanggung Jawab untuk Masa Depan yang Lebih Hijau

# AWS Layanan utama

- Amazon Aurora
- Amazon DynamoDB
- Amazon DocumentDB (dengan kompatibilitas MongoDB)
- Amazon ElastiCache
- Amazon S3
- Kelas Penyimpanan Amazon S3
- Kelas penyimpanan Tingkat Cerdas Amazon S3
- Kelas penyimpanan Amazon S3 Express One Zone
- Amazon Elastic Block Store
- AWS Lambda
- Layanan Kontainer Elastis Amazon
- Layanan Amazon Elastic Kubernetes
- Amazon GameLift
- AWS Compute Optimizer

# Kesimpulan

Game dirancang untuk memberikan pengalaman hiburan kepada khalayak pemain global dan memiliki karakteristik penggunaan yang biasanya tidak dapat diprediksi dan bervariasi. Game Industry Lens menjelaskan jenis skenario umum yang biasanya terdiri dari arsitektur game dan menyediakan serangkaian pertanyaan dan praktik terbaik untuk dipertimbangkan saat Anda membangun dan mengoperasikan game di cloud. Dengan menerapkan kerangka kerja ini ke arsitektur game Anda, Anda dapat membangun game yang andal, aman, efisien, dan hemat biaya di cloud.

# Kontributor

Individu-individu berikut berkontribusi pada dokumen ini:

- Adam Hatfield, Arsitek Solusi Senior, Amazon Web Services
- Brady Webb, Manajer Akun Teknis, Amazon Web Services
- Bruce Ross — Arsitek Solusi Senior, Pemimpin Lensa Well-Architected, Amazon Web Services
- Caleb Cecil, Arsitek Solusi Asosiasi, Amazon Web Services
- Carlos Perez, Sukses Optimasi Cloud SA, Amazon Web Services
- Chase Herrington, Manajer Akun Teknis ESL, Amazon Web Services.
- Chris Blackwell, Arsitek Solusi Sr., Amazon Web Services
- Corey Ouderkirk, Manajer Akun Teknis Sr., Amazon Web Services
- Derek Villavicencio, Prototyping SA, Amazon Web Services
- Erik Ynigo Becerril, Arsitek Solusi Senior, Amazon Web Services
- Grzegorz Ochmanski, Arsitek Solusi Sr., Amazon Web Services
- Hadrian Baron, Manajer Akun Teknis Sr., Amazon Web Services
- Ian Armbruster, Manajer Solusi Pelanggan Sr., Amazon Web Services
- Jed Obray, Manajer Akun Teknis Sr., Amazon Web Services
- Khurram Khokhar, Manajer Akun Teknis Sr., Amazon Web Services
- Kyle Somers, Manajer Sr., Arsitektur Solusi, Amazon Web Services
- Madhuri Srinivasan, Penulis Teknis Senior, Well-Architected, Amazon Web Services
- Matthew Wygant, Bimbingan Sr. TPM, Well-Architected, Amazon Web Services
- Nataliya Godunok, Sukses Pengoptimalan Cloud SA, Amazon Web Services
- Nirav Doshi, Arsitek Solusi Utama, Amazon Web Services
- Olivia Liddell, Arsitek Solusi, Amazon Web Services
- Randy James, Manajer Akun Teknis Utama, Amazon Web Services
- Reou Ando, Arsitek Solusi Game, Amazon Web Services
- Richard Raseley, Manajer Akun Teknis Sr., Amazon Web Services
- Sam Patzer, Arsitek Solusi Senior, Amazon Web Services
- Scott Selinger, Arsitek Solusi Sr., Amazon Web Services
- Sean Allen, Arsitek Solusi Sr., Amazon Web Services

- Serge Poueme, Arsitek Solusi Senior, Amazon Web Services
- Stewart Matzek, Penulis Teknis Senior, Well-Architected, Amazon Web Services
- Trenton Potgieter, Arsitek Solusi Sr, Amazon Web Services AI/ML/Analytics

# Revisi dokumen

Untuk mengetahui jika ada perubahan pada laporan resmi ini, Anda dapat berlangganan umpan RSS.

| Perubahan | Deskripsi | Tanggal |
| --- | --- | --- |
| Versi lensa baru | Seluruh lensa diperbarui dengan panduan praktik terbaik yang baru. | Desember 9, 2025 |
| Publikasi awal | Whitepaper pertama kali diterbitkan. | November 19, 2021 |

# AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.