Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Videó streaming szolgáltatások implementációja

Diplomaterv

Készítette
Piller Trisztán

Konzulens
Kövesdán Gábor

2024. december 13.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Piller Trisztán*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2024. december 13.

Piller Trisztán
hallgató

Kivonat

Az IT szakma meghatározó kihívása a magasan teljesítőképes, könnyen skálázódó és stabil infrastruktúra létesítése különféle üzleti célok megvalósítására. A hírközlés, a média és a szórakoztatatás iparágaiban is kiemelt figyelmet kap ez a kihívás.

Ezek a virágzó és feltörekvő iparágak folyamatosan igénylik az olyan szoftverfejlesztőket, akik ezekre az iparágakra specializáltan is folyamatosan képzik magukat, valamint mélyen ismerik a szakterület technológiáit.

Diplomatervem célja demonstrálni egy az Amazon Web Services platformján futó felhő alapú médiaszolgáltatás-rendszer alapos tervezésének, implementálásának, valamint tesztelésének folyamatát. A rendszer a videó streaming szolgáltatások területén nyújt weben elérhető megoldást, és a felhasználók számára lehetővé teszi, hogy saját időbeosztásuknak megfelelően férjenek hozzá videótartalmakhoz (Video-on-Demand), valami élő adásokat is tudjanak megtekinteni (live streaming).

A megoldás ismertetése során kiemelt fókuszt kap a komponensek közötti laza kapcsolás kialakítása, az IT biztonsági kockázatok kezelése, az konfigurációmenedzsment fenntarthatósága. Felhasználásra kerülnek modern webes technológiák és DevOps technikák, mint a konténerizáció, a serverless függvények, a CI/CD csővezetékek és az Infrastructure as Code.

Abstract

A key challenge for the IT profession is to build highly performant, easily scalable and stable infrastructure to support a variety of business goals. This challenge is also a major focus in Telecommunications, also in the Media & Entertainment industry.

These booming and emerging industries are in constant need of software developers who train themselves continuously for these industries and have a deep knowledge of the technologies in the field.

My thesis project aims to demonstrate the process of thoroughly designing, implementing and testing a cloud-based media delivery system running on the Amazon Web Services platform. The system will provide a web-based solution for video streaming services, allowing users to access Video-on-Demand content and live streams.

The solution will focus on the design of loose coupling between components, the management of IT security risks and the sustainability of configuration management. Modern web technologies and DevOps techniques such as containerisation, serverless functions, CI/CD pipelines and Infrastructure as Code will be used.

1. fejezet

Bevezetés

1.1. A témaválasztás indoklása

Arról írok, hogy miért választottam a témát: Érdemesnek tartom a szakterület vizsgálatát, a média streaming maga elég szűk területe a szoftveres világnak, mégis roppant nagy szaktudást lehet benne felvenni. Médiaszolgáltatások fejlesztésének infrastrukturális és szoftveres igényei jelentős kihívást jelentenek a szakma architektjei számára, és a témában való elmélyülés nagyban hozzájárulhat a szakmai tudásom bővítéséhez.

1.2. Kihívások a tématerületen

Ez a szekció arról fog szólni, hogy szoftver architekteknek milyen kihívásokkal kell szembenézniük, ha egy olyan kaliberű szórakoztató médiaszolgáltatás-rendszert kell megtervezniük, mint a Netflix vagy a Twitch.

Fel lesz sorolva jó pár dolog (felhasználói élmény szintre emelése, gyorsan fejlődő protokollok, különféle hardverekre/platformokra való optimalizálás, globális forgalmazás), kiemelve hogy én a biztonságra, a skálázhatóságra, az alaposan megtervezett és fenntartott infrastruktúrára helyezem a hangsúlyt.

2. fejezet

Technológiai áttekintés

2.1. A videó streaming protokolljai

Itt jó sok alszekcióban kifejtem a legfontosabb end-to-end protokollcsaládjait a streamingnek. Szó lesz az RTP/RTMP (és annak pull és push változatáról), a HLS/-DASH, a WebRTC protokollokról. Ezek kardinalitása (források és felhasználók számát is figyelembe véve!) és a különféle alkalmazási területeik lesznek a fókuszban. Felmerül, hol működik az ABR (Adaptive Bitrate Streaming).

2.2. Amazon Web Services

Bevezető az AWS szolgáltatások világába, röviden bemutatom, hogy épül fel egy AWS account, milyen szolgáltatáscsoportokat ajánl, mivel lehet korlátozni a hozzáférést, az erőforrások akár egymás közötti kommunikációjá irányítani.

2.2.1. Hagyományos webalkalmazások erőforrásai AWS-en

Cloudfront mint CDN, S3 mint objektumtároló, RDS mint relációs adatbázis, Lambda mint szerver nélküli függvény, VPC mint privát hálózat, IAM mint hozzáféréskezelés, WAF mint web application firewall, Route53 mint DNS szolgáltatás.

2.2.2. Videó streaming AWS-en

Ebben pedig a már konkrét felhasznált (és nem felhasznált) AWS szolgáltatásokat mutatom be, mint a MediaLive, MediaPackage, és melyiket miért választottam, és a többit (IVS, saját konténerizált ffmpeg szerver) miért nem.

2.3. A webes komponensek technológiái

Ebben a részben a konkrét AWS-beli PaaS és FaaS szolgáltatásokon futó több rétegű appok választott technológiáit mutatom be: Node.js, React, Prisma, Postgres.

2.3.1. TypeScript és JavaScript nyelvek

Miért választottam a TypeScriptet, miért jobb a JavaScriptnél, felhasználása a JavaScriptnek a Lambda és Cloudfront Function-ökben.

2.3.2. Node.js ökoszisztéma

Node.js futtatókörnyezet, NestJS keretrendszer, Prisma ORM, AWS SDK.

2.3.3. React és társkönyvtárai

React, React Hook Forms, React Query, Vite.js, TailwindCSS.

2.3.4. PostgreSQL

Hasznos, mert open source, sok helyen elérhető DBMS motor.

3. fejezet

A tervezett felhőarchitektúra

Gyors bevezető arról, hogy külön AWS accountba dolgoztam és felügyeltem a költségeim.

3.1. Követelmények

Funkcionális és nem-funkcionális követelmények, a rendszerrel szemben támasztott elvárások. Event driven architektúra (kiterjeszthetőség érdekében), skálázhatóság, biztonság, megbízhatóság.

3.2. Logikai felépítés

High-level leírása az AWS accounton belüli/datacenteren belüli hálózati kommunikáció rétegeinek.

3.3. Video-on-Demand kiszolgálás folyamata

High-level leírása a VoD kiszolgálás folyamatának, megemlítve, melyik választott szolgáltatások vesznek részt ebben.

3.4. Live streaming folyamata

High-level leírása a live streaming folyamatának, megemlítve, melyik választott szolgáltatások vesznek részt ebben.

3.5. Konfigurációmenedzsment

Terraform választásának indoklása. Kialakított Terragrunt module rendszer. GitHub actions a Terraform tervek előnézetére, OIDC felállítása az AWS role felvételére.

4. fejezet

Kliens közeli komponensek implementációja

4.1. A CDN és a hozzácsatolt erőforrások

A CDN és az originjeinek tárgyalása, melyik origin mire való. VPC Origin tárgyalása, annak haszna. A WAF szerepe, a felkapcsolt cert és WAF web ACL szerepe.

4.2. A statikus weboldal

Az S3 bucketee. A statikus website kiszolgálásának módja.

4.2.1. A React alkalmazás fejlesztése

Fejlesztés részletei. Nem annyira lényeges most ebben a dolgozatban, de néhány szép kihívást jelentő kidolgozott form és egyebeket be lehet itt mutatni.

4.2.2. A weboldal telepítésének CI/CD folyamata

GitHub Actions a smoke tesztekre, workflowk, a deploymentek. Értsd itt: S3 telepítés.

4.3. Média erőforrások objektumtárolói

Hogy lettek felkonfigurálva és miért az egyes S3 bucket-ok (bucket policy, CORS policy).

4.4. Elemental MediaLive és MediaPackage a live streamingben

Az Elemental stack részeinek felkonfigurálása, a MediaLive channel és a MediaPackage channel felépítése, a MediaPackage endpoint konfigurálása. Miként kerül kiszolgálásra, melyiket mire használom. OBS bekötésének módja.

5. fejezet

Szerver oldali folyamatok implementációi

5.1. A virtuális privát felhő komponensei

A VPC-beli (Virtual Private Cloud) subnetek, a security groupok, route táblák. A biztonság vizsgálata.

5.2. A Node.js alkalmazás fejlesztése

Fejlesztés részletei. Kitérve arra, hogy miképp könnyíti a munkát a Prisma, milyen egyéb szolgáltatások kerültek be, mi a felépítése a reponak, miért választottam ezt a stacket. S3 bucketba való mentése a videónak egy érdekes rész. Itt lehet szó az adatbázisbeli entitásokról is.

5.3. A konténerizált környezet

Leírás, hogy miért választottam a konténerizált környezetet, a konténerizálás előnyeit, hátrányait. Hogy használható ki a legjobban a konténerizáció az Application Load Balancer-rel együtt. Miképp kapcsoltam ezt a kettőt össze (ECS service, ALB).

5.3.1. A Node.js szerveralkalmazás ECS-en

Az ECS orkesztrációs toolsetjének kialakítása, a konténer rétegződés felépítése ECS-ben, a konténer registry (ECR) bekötése. Környezeti változók, portok, ALB-re való kötése. Miből állt a dockerizálás nekem (Dockerfile, registry, image build, push, networking).

Milyen IAM role-okat kellett feltenni rá, mikkel kommunikál kifelé, mi indokolta, hogy publikus subnetbe kerüljön. Hogy hív meg más külső rácsatlakozó erőforrásokat (S3 bucket, RDS instance, Lambda függvény, MediaLive channel).

5.3.2. A szerveralkalmazás CI/CD folyamatai

GitHub Actions a smoke tesztre, a deploymentre: Docker build és ECR-be telepítés.

5.4. Elemental MediaConvert felhasználása

Az Elemental MediaConvert API használata a Lambdából, illetve hogy hogy hívódik meg a Lambda, milyen triggerrel, milyen környezeti változókkal, milyen IAM roleal. Hogy kellett felkonfigurálni a MediaConvert job, hogy kellett magát a Lambdát felkonfigolni, hogy tudja is hívni.

6. fejezet

Tesztelés és mérés

6.1. Az infrastruktúra terhelése

Szimpla load tesztelés összeállításáról fogok itt értekezni, és a mérési eredmények kiértékelése.

6.2. Népszerű szolgáltatók metrikái

Keresni kell cikkeket Netflix blogban vagy valahol arról, a népszerű szolgáltatók milyen SLA-val, milyen statisztikával dolgoznak.

7. fejezet

Összegzés

Tanulságok, a rendszer működésének és fejlesztési élmények értékelése. Média streaming jövője saját meglátások szerint.

7.1. Továbbfejlesztés lehetőségei

Min lehetne javítani, mikroszolgáltatásos architektúra stb.

7.1.1. Vendor lock-in kezelése

Miként befolyásolja egy üzlet működését a vendor lock-in, és hogyan lehet ezt kezelni. Miképp lehetne a jövőben a vendor lock-in-t csökkenteni ebben a rendszerben.

7.2. Kitekintés: egyéb AWS szolgáltatások

Miért nem használtam az Aurorat RDS helyett, miért nem került sor Amplify Hosting alkalmazására a frontend és backend összekapcsolására és egy stackben való deployálására (skálázhatóságra való kiélezés miatt).