Skálázás a gyakorlatban

irányelvek, stratégiák - Amazon szolgáltatásokon keresztül

Bencs Balázs
Senior Software Developer

balazs.bencs@attrecto.com

+36 70 682 72 72

Mi a skálázás?

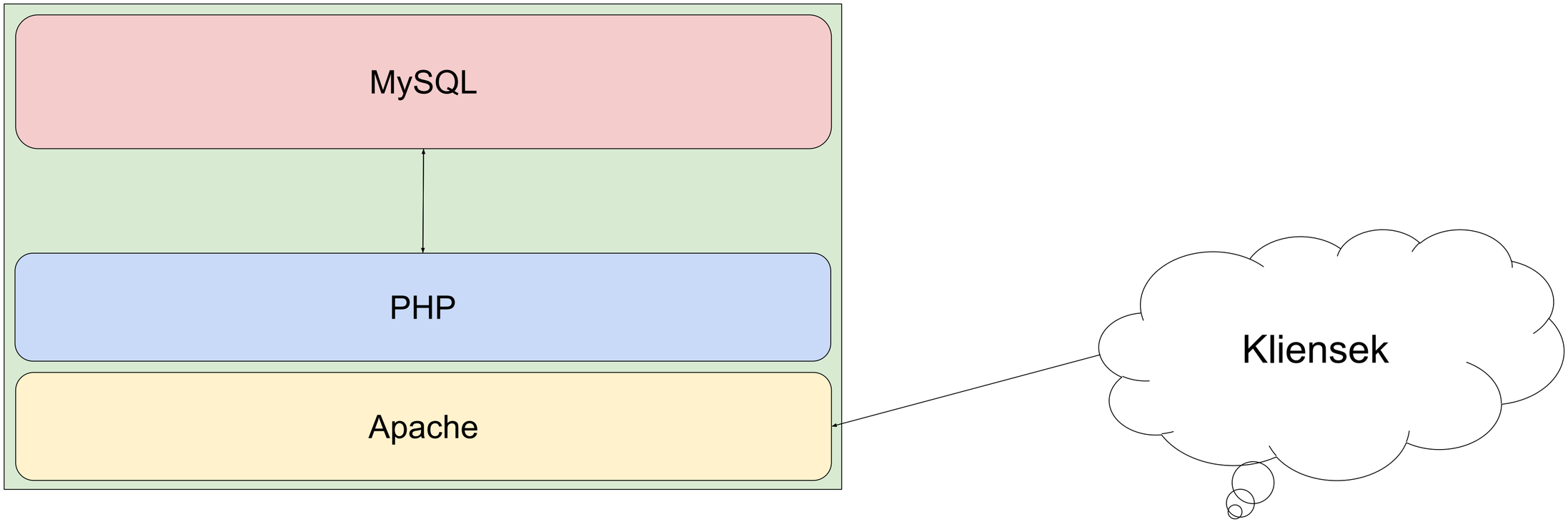
Redundancia, vagy performancia?

Megtartani az alkalmazás működésének
a színvonalát magas terhelés mellett is,
ahogy a felhasználók száma növekedik.

A kezdetek

Miből is áll egy webszolgáltatás

- Valószínüleg egy egyszerű LAMP stack, vagy hasonló - 1 szerveren az alkalmazáskiszolgáló, az adatbázis és minden egyéb.



Főbb kulcs pontok

- A fő irányelv legyen az, hogy fogod a különböző komponenseket, és szétválasztod őket. Ezzel a komponensek külön-külön könnyebben skálázhatóak és leállás esetén is a külön lévő komponensek állnak csak le - plusz így micro-szolgáltatásokat is könnyebben létre tudsz hozni.
- Csak olyan feladatokba érdemes időt tenni, ami megkülönböztet business oldalon másoktól, ne találd fel a spanyol viaszt.
- A skálázás és redundancia két külön dolog, de gyakran együtt jár.

Amazon alapok

- 12 AWS régió van a Földön.
- A régió egy fizikai hely a földön, ahol az Amazonnak van több Availability Zone-ja.
- Egy Availability Zone (AZ) általában egy különálló datacenter, bár állhat több datacenterből is.
- Minden AZ-nek külön áram ellátása és hálózati kapcsolata van, teljesen különállóak.
- Az egyetlen kapcsolat az AZ-k között az egy low latency network. Az AZ-k lehetnek például 5 vagy 15 kilométer távolságban, de a hálózat köztük olyan gyors, hogy úgy viselkednek, mint egyetlen nagy datacenter.
- Minden régióban legalább 2 datacenter van, összesen 32 a Földön.
- AZ-ket használva, könnyen lehet skálázható alkalmazásokat építeni.
- 2016-ban legalább 9 új AZ nyílik majd meg, és 4 új régió fog létesülni.

Amazon alapok

- Az AWS-nek 53 “edge location”-je van a Földön.
- Ezeket az Edge location-öket használja a CloudFront, az Amazon’s Content Distribution Network (CDN) és a Route53 (az Amazon menedzselhető DNS szervere)
- Az Edge location-ökkel a felhasználók tartalmakat érhetnek el nagyon kis késleltetéssel, akárhol is legyenek a Földön.

Block Service-ek építése

- Az AWS különböző szolgáltatásokat épített, amik több AZ-t használva magas rendelkezésre állást és hibatűrő működést biztosítanak.
- Ezeket díjak ellenében lehet használni anélkül, hogy saját magadnak kelljen az infrastruktúrát kialakítani.
- Néhány szolgáltatás ami elérhető egy AZ-n: CloudFront, Route 53, S3, DynamoDB, Elastic Load Balancing, EFS, Lambda, SQS, SNS, SES, SWF.
- Tehát magas rendelkezésre állású architektúra hozható létre ezeknek a szolgáltatásoknak a használatával egy AZ-n belül is akár.

LOAD SZINTEK

attrecto
smartphone solutions

1 felhasználó

- Ebben az esetben te vagy az egyetlen felhasználó, aki használja az alkalmazást
- Valahogy így néz ki az architektúrád:
  - Egy single instance, talán egy **t2.micro**, vagy picivel nagyobb.
- Ezen az egy instance-on fut a teljes web stack: webapp, DB, egyéb management dolgok.
- Talán használsz Amazon Route 53-mat a DNSre, vagy van egy domain szolgáltatód.
- Van egy darab Elastic IP címed az instance-hoz.
- Teljesen jól működik - egy darabig!

Vertikális skálázás

- Gyorsabb szolgáltatásra vágysz. A legegyszerűbb, hogy fogsz egy nagyobb instance-t. Talán egy c4.8xlarge vagy m3.2xlarge példaképp.
- Ezt hívjuk vertikális skálázásnak.
- Csak megállítod a jelenlegi instance-t, választasz egy nagyobbat és újraindítod, bármilyen gondra tudsz nagyobb gépet választani.
- Néhány Amazon service kínál Provisioned IOPS opciót, ami garantálja a jó sebességet. Így megoldható egy kisebb instance-al és például Amazon DynamoDB-vel, hogy skálázott szolgáltatást használhass és akkor nem neked kell megépítened.
- Van evvel egy gond: nincs failover, nincs redundancia. Ha az instance-ban gond van, meghal az egész webapp, és az instance-od csak nőni fog, valami mást kell kitalálni.

> 10 felhasználó

- Különítsd el az egyetlen host-ot külön instance-okra.
- Egy host az alkalmazásnak
- Egy host az adatbázisnak
- A külön hostok biztosítják, hogy az alkalmazásod skálázható legyen szeparáltan:

  Mondjuk egy nagyobb instance kell a DB-nek, de egy kissebb elég a webappnak.
- Vagy saját adatbázis helyett használhatnál külső szolgáltatást.
- Egy database admin vagy? Backupokkal szeretnél törődni? Magas rendelkezésre állással mi a helyzet? Patch-ekkel? Operációs rendszer választás/konfiguráció/felügyelet?

> 10 felhasználó

- Ezeknek az előnye, hogy multi AZ-s adatbázist építhetsz egy klikkel.

  Nem kell gondolni a replikációra, vagy hasonlókra. Az adatbázis magas rendelkezésre állású lesz és megbízható.
- Létezik ebből sok féle: Amazon RDS (Relational Database Service). Ebben van Microsoft SQL Server, Oracle, MySQL, PostgreSQL, MariaDB, Amazon Aurora, Amazon DynamoDB (NoSQL).
  - Amazon Aurora: Automatikus storage skálázás 64TB-ig. Nem kell ezzel se külön törődnöd.
  - 15 read-replica telepítése lehetséges
  - Inkrementális biztonsági mentés S3-ra.
  - 6 utas replikáció 3 AZ-n keresztül. Ezzel hibatűrővé is vált minden.
  - MySQL kompatibilis

> 10 felhasználó

- NoSQL helyett kezdj SQL adatbázissal
- A technológia jól bejáratott
- Rengeteg már létező kód van, nagy közösség, support, könyvek, tool-ok
- Nem lesz gondod az SQL-el az első 10 millió felhasználóig legalább. (Kivéve ha hatalmas adattömeget kell kezelned)
- Tiszta és egyértelmű metódusok vannak a skálázására.

> NoSQL

- Mikor van szükséged NoSQL-re?
- Ha több mint 5TB-nyi adatot kell tárolnod évente, vagy ha nagyon intenzív az adatmozgás az adatbázisodban (sok az írási művelet).
- Ha az alkalmazásodnak szuper-alacsony késleltetési követelményei vannak
- Ha tényleg nagyon nagy adatmennyiség megy keresztül az adatbázison. Ilyenkor érdemes erősen tweak-elni az IO műveleteket írásra és olvasásra is.
- Nincs semennyi kapcsolat az adataid között az adatbázisodban (relációk)

> 100 felhasználó

- Egy különálló host kell a webalkalmazásnak
- Amazon RDS-ben tárold az adatbázist, ezzel mindent elintéztél
- Nagyjából ennyi az egész, itt még nincs vészes forgalom

> 1000 felhasználó - kezd érdekesebb lenni

- Jelenleg 1 host szolgálja ki az appot, ha az lehal, minden megáll - rendelkezésre állási gondjaid vannak.
- Szóval kell még egy webapp host egy másik AZ-n belül, ami oké,
  mert alacsony a latency az AZ-k között.
- Valószínüleg kelleni fog egy slave adatbázis az RDS-en, ami egy másik AZ-ban fut.
  Ha valami gond lenne a masteren, átkapcsolhat a rendszer a slave-re. Az alkalmazásodban nem kell változtatni semmit, ugyanaz az endpoint fogja kiszolgálni.
- Egy Elastic Load Balancer (ELB) kell neked, ami elosztja a terhelést a két webhostod között a két AZ-ben.

> 1000 felhasználó - kezd érdekesebb lenni

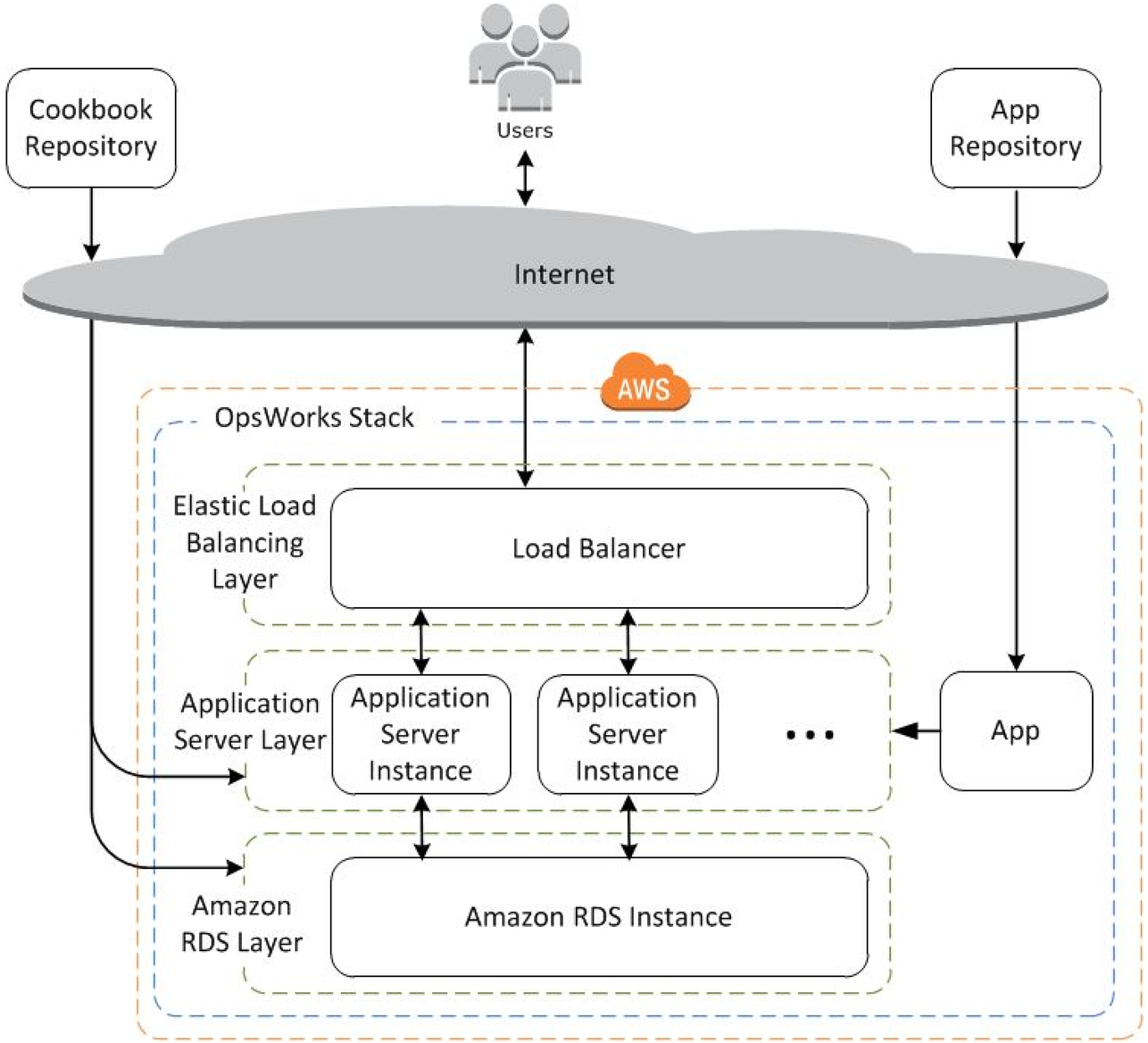
Elastic Load Balancer (ELB)

- Az ELB egy magas rendelkezésre állású load balancer. Minden AZ-ban elérhető, gyakorlatilag egy sima DNS végpont az alkalmazásodhoz. Csak be kell tenni a Route 53-ba és azonnal osztja is a forgalmat a web host-jaid között.
- Az ELB-ben van Health Check, hogy biztosan ne menjen halott host felé a forgalom.
- Mindent skáláz, anélkül, hogy bármit is kellene csinálnod. Ha plusz forgalmat észlel, akkor a háttérben horizontálisan és vertikálisan is tud skálázni!
- Nem kell külön menedzselned!

> 10 000 - 100 000 - durvul a helyzet

- Az előző konfigurációban volt 2 instance egy ELB mögött, de a gyakorlatban lehet akár több 1000 instance-od is az ELB mögött. Ez a horizontális skálázás.
- Több Read-replikát kell hozzáadnod az adatbázisodhoz, ez leveszi a masterről a terhet, így ő törődhet csak az írással.
- Érdemes megfontolni, hogy hogyan lehet levenni terhet a web host-jaidról, mondjuk úgy hogy a statikus tartalmat szolgálja ki az Amazon S3 vagy az Amazon CloudFront. A CloudFront az Amazon saját CDN storage szolgáltatása ami az 53 edge location-ön mindenhol egyszerre tárolja az adatokat a világon.

> 10 000 - 100 000 - durvul a helyzet



GYŐR DEVUP

SZOLGÁLTATÁSOK

attrecto
smartphone solutions

Amazon S3

- Az Amazon S3 egy object alapú storage.
- Nem olyan mint az EBS, nem olyan storage ami egy EC2-es instance-hoz csatlakozik, ez object storage, nem block storage.
- Nagyon jól használható statikus tartalomra: javascript, css, képek, videók…
- Ez leveszi a web host-odról a terhet.
- Nagyon megbízható, 11 9-es a rendelkezésre állás.
- A végtelenségig skálázható, van aki több Petabyte adatot tárol benne.
- Az objektumok mérete 5TB-ig mehet fel.
- Támogatott a titkosítás is - használhatod az Amazon-ét, sajátot, külső titkosítási service-t...

CloudFront

- Az Amazon CloudFront cache-eli a tartalmat az edge location-ökön, hogy a lehető legkisebb latency legyen elérhető bárhol a Földön.
- Egy jó CDN nélkül a felhasználóid magas latency-t fognak észlelni, illetve a szervereid nagy load alá kerülnek, mert tartalmat is kiszolgálnak, és az adat-request-eket is kezelik - sok meló egyszerre.
- Már megtörtént sztori volt hogy egy ügyfélnek 60-70 Gbps kellett a kiszolgáláshoz.
  A web host azt se tudta mi van, annyira le volt terhelve a webszerver.
- A CloudFront-al minden megjavult egyszerre.

Caching

- Tudod még enyhíteni a loadot úgy, hogy a session kezelést kívülre helyezed
- Tárold a sessiont ElastiCache-ben, vagy a DynamoDB-ben (alternatíva).
- Ezzel a megoldással a rendszeredet egy későbbi lehetséges automatikus skálázáshoz is elő tudod készíteni.
- Sőt még tovább enyhítheted a loadot, ha cacheled az adatokat az adatbázisból az ElastiCache-el.
- Nem kell minden SELECT-nek a DB-n futni, a cache elintézi, amíg a DB-d fontosabb dolgokkal is tud foglalkozni (írási műveletek).

noSQL - Amazon DynamoDB

- Ez egy menedzselhető NoSQL adatbázis
- A NoSQL gyors és kiszámítható sebességű
- Ő egy kulcs-érték storage, ahol a JSON formátum is támogatott
- A maximális dokumentum méret 400kB
- Te rendelkezel a forgalmaddal, csak rajtad áll mekkorák lesznek az adat csomagok
- Ezzel nagy mértékben tudod befolyásolni az írási és olvasási sebességet
- Teljesen redundáns és nagy hibatűrő képességű
- Több AZ-ben is elérhető

ElastiCache

- Az Amazon ElastiCache egy menedzselhető Memcached vagy Redis instance
- Egy saját memcached cluster-t menedzselni nem kis móka, időigényes, az Amazon elintézi ezt helyetted.
- Automatikusan skálázódik, illetve ha lehal egy instance, automatikusan létrejön egy új helyette - így teljesen hibatűrő.
- Még tudod a load-ot csökkenteni, ha a dinamikus tartalom egy részét átteszed a CloudFront-ra.
- Sokan gondolják, hogy csak JS-re, CSS-re, meg ilyenekre jó, de nem csak: https://aws.amazon.com/cloudfront/dynamic-content/

Automata skálázás

- Ha elég vasat felhalmoztál arra, hogy minden kiugró load-ot ki tudj szolgálni, akkor égeted a pénzt a kezeid alatt. Jobb lenne inkább egy olyan cluster, ami automatikusan nő a megnőtt tartalommal, mondjuk egy Black Friday-en.
- Beállíthatod a minimum és maximum méretét a pool-odnak, így beállíthatod hogy legalább X instance fusson, de Y-nál ne legyen több új.
- Ebben segít a CloudWatch management service, ami minden alkalmazásodba integrálódik.
- A CloudWatch események triggerelhetik a skálázási eseményeket.
- CPU használaton skálázol, vagy a latency nem jó? Esetleg a hálózati forgalom nem megfelelő? Saját metrikát is állíthatsz a CloudWatch-ban, így ha valami app specifikus dolgot szeretnél állítani automatikusan, akkor arra a metrikára figyeli az eseményeket és triggereli a skálázást

Beanstalk

- Az Amazon Web Services (AWS) több tucat szolgáltatást kínál, amelyek minden egyes külön funkcióra kínálnak megoldást. Amíg ezek a szolgáltatások flexibilitást kínálnak és így szabad kezet ahhoz, hogy saját magad alakítsd ki az infrastruktúrád - lássuk be, hogy ebben azért elég sok kihívás van.
- Az AWS Elastic Beanstalk-al nagyon gyorsan létrehozhatsz, menedzselhetsz alkalmazásokat az AWS-en, anélkül hogy aggódnod kéne az infrastruktúra miatt - megkönnyíti az instance-ok létrehozását illetve skálázását. Csak fel kell töltened az alkalmazásodat az Elastic Beanstalk-ra és automatikusan elintézi neked a rendelkezésre állást, terhelés elosztást, skálázást, deployment processzt, illetve az alkalmazás monitorozását (health check).

> 500 000+ felhasználó

- Az előző konfigurációhoz hozzájönnek az auto scaling group-ok.
  Az auto scaling group legalább kettő AZ-ban, vagy többen is létezik,
  de egy régión belül kell hogy legyenek.
- Például van 3 webes instance-od mindegyik AZ-ban (ez a szám lehet akár több 1000 is).
  Ilyenkor meg tudod adni, hogy legalább 3 instance-ot szeretnél az AZ-kban, de maximum 1000-et.
- ElastiCache-t kell használni, hogy a legtöbbet futó SELECT-eket az adatbázisból cache-eld le.
- DynamoDB-t használni, hogy a session adatokat tárold külön
- Muszáj lesz valami monitoring stratégia, metrikák készítése, illetve masszívabb loggolás

> 500 000+ felhasználó

- Host szinten kellenek majd metrikák, hogy meg tudd figyelni egy instance-on belül egyetlen CPU foglaltságát az autoscaling group-on belül is. Különben nem fogod tudni, hogy hol a gond a rendszerben.
- Metrika kell a Load Balancer-en is, hogy legyen információd arról, hogy egy csoportnyi instance-nak milyen a teljesítménye.
- Log analízis: Átlátható log analizátor kell (pl.: CloudWatch, Bugsnag, New Relic, Sentry)
- Teljesítmény analízis kliens oldalról. Muszáj tudnod, hogy milyen teljesítményt nyújt az appod. Itt lehet használni pár enterprise tool-t: New Relic vagy Pingdom.
- Tudnod kell hogy a felhasználók lassúnak látják-e az oldalad, vagy sok a hiba?
- Amennyit lehet konfigurációval nyerni a teljesítményen, azt vesd be. Az Auto Scaling segít, nem jó ha egy rakás instance csak mondjuk 20%-osan van kihasználva.

Elindítod az új Twittert! (kb. 10 millió user)

- Ahogy kezd egyre nagyobb lenni az adatáramlás, gond lesz az adatbázissal, főleg amiatt, hogy az írási feladatok egy szint után nem lesznek gyorsabbak
- Egyes funkciókat más típusú DB-kre kell áttenni (NoSQL, graph, stb..)
- Federation - Több adatbázisra kell felosztani a funkciók alapján
  - pl: fórum adatbázis, user adatbázis, termékek adatbázis..
  - Ezeket külön-külön is lehet skálázni
  - Negatívum: nehéz cross-DB lekérdezéseket csinálni, és nem a legjobb megoldás, ha át akarsz térni a ...
- Sharding - egy dataset felosztása több host-ra
  - Komplexebb az alkalmazás rétegben, de nincs gyakorlati limit a skálázhatóságán
  - Például a userek egy harmada az egyik shard-on, és a másik két harmad két másik shard-on lesz

Összegzés

- Csak olyan feladatokba ölj időt és pénzt, ami megkülönbözteti az alkalmazásodat egy hasonló megoldástól. Már létező és működő szolgáltatásokkal könnyebb dolgod lesz.
- Az Amazon-nak rengeteg szolgáltatása van ami alapértelmezetten hibatűrő, mert több AZ-ben terülnek el. Példa: e-mail, adatbázisok, monitoring, metrikák, keresés, task queue, transcoding, logging, és persze a compute.
- Ne építsd meg magad, ha már létezik!
- Használj microservice-eket: fogd az alkalmazásodnak a komponenseit és darabold szét részekre (nem csak web és adatbázis), a különálló szolgáltatásokat könnyebb külön-külön skálázni, mint egy nagy alkalmazást.

Ők használják

NETFLIX

Adobe

GYŐR DEVUP

airbnb

slack

Pinterest

reddit

yelp

Köszönöm a figyelmet!

attrecto
smartphone solutions