DevOps, 3. časť: Dostupnosť služby, CDN, zálohy, CI/CD, testovanie aplikácií (TDD), bezpečnosť

adam.puskas@uxtweak.com

Vývoj progresívnych webových aplikácií

Lektor: Ing. Adam Puškáš

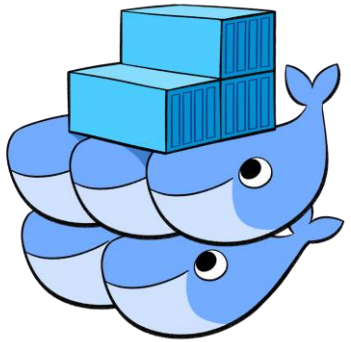
Vedúci kurzu: Ing. Eduard Kuric, PhD.

27.4.2022

Na minulej prednáške...

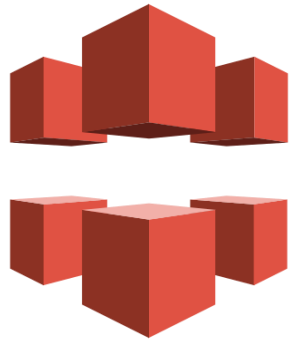
- Cloud computing - pojem, vlastnosti, typológia, kontexty využitia
  - Agilita, elasticita, abstrakcia, samoobsluha, cenová flexibilita
  - Public vs. private vs. on-premise cloud
  - SaaS vs. PaaS vs. IaaS

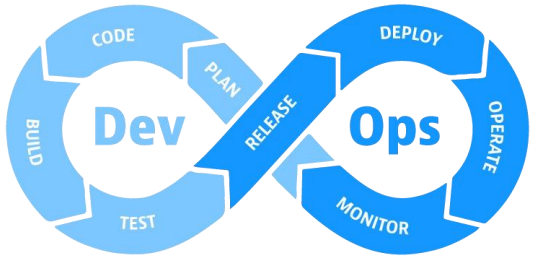
- Orchestrácia SW kontajnerov - Docker Swarm, AWS ECS, Kubernetes
- Cloud typu IaaS Amazon Web Services (AWS):
  - Prehľad služieb
  - Nasadenie služby na AWS ECS - príklad



Agenda dnešnej prednášky

- Dostupnosť služby a viaczónové nasadenie (multi-AZ)
- Optimalizácia doručovania obsahu (CDN) a object storage
  - AWS CloudFront + AWS S3
- Replikácia a zálohovanie dát
- DevOps ako pojem, CI/CD, správa verzií (Git), automatizácia (Jenkins)
- Testovanie aplikácií, TDD, základy bezpečnosti - penetračné testovanie





Dostupnosť služby a viaczónové nasadenie (multi-AZ)

- **Vysoká dostupnosť** služby (high availability)
- Dôsledky prevádzky služby s nedostatočnou dostupnosťou:
  - Odliv klientov (špeciálne B2B - plnenie zákaziek, napr. agentúry)
  - Šírenie zlého povedomia - strata ďalších existujúcich i potenciálnych klientov
  - Faktická nemožnosť presadiť sa na globálnom trhu (SLA, konkurencia)

- Úskalie vysoko dostupnej architektúry - cena
  - Problém najmä pre malé tímy (startupy)
  - V prípade úspechu sa počiatočná investícia mnohonásobne vráti

Vysoko dostupná služba - vlastnosti

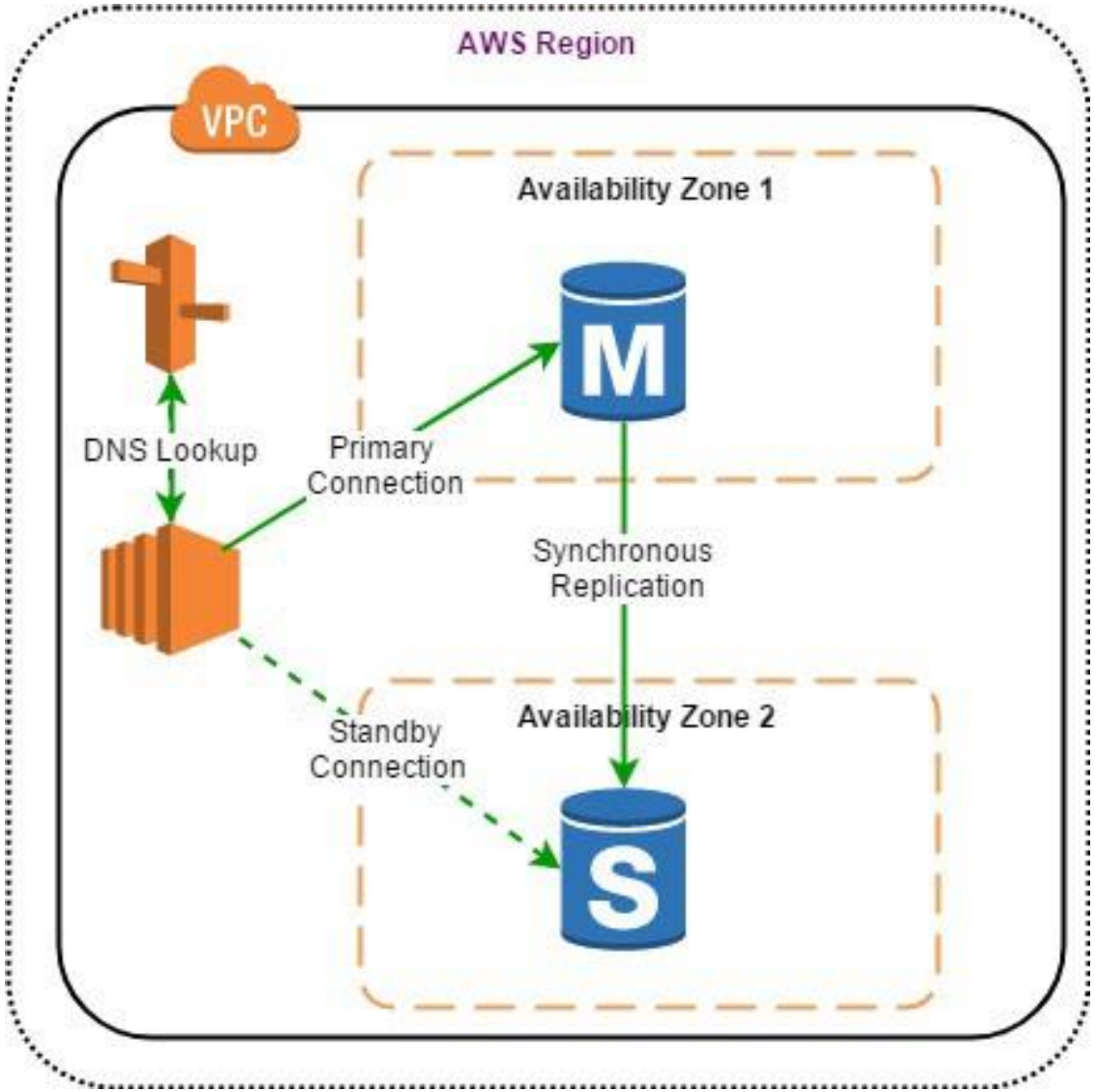
- Redundancia (redundancy)
  - Dôležité komponenty systému majú viacero **replík** (dáta, funkcionalita)
  - Repliky preberú úlohu primárnych inštancií v prípade ich zlyhania (potreby)
  - Repliky sú typicky umiestnené v rôznych regiónoch (zónach dostupnosti - AZ)
- Monitorovanie (monitoring)
  - Systém a jeho komponenty sú pokryté **metrikami**, **logmi**, **alarmami**
  - Vždy je známy **stav** jednotlivých služieb, ako aj systému ako celku
  - V prípade potreby možno na zmenu stavu reflektovať akciou (napr. spustenie novej repliky služby)

Vysoko dostupná služba - vlastnosti /2

- Rýchle prepnutie (failover)
  - Komponent systému - aktívnu (primárnu) inštanciu komponentu je možné **okamžite prepnúť na sekundárnu**
  - Napr. relačná databáza - udržiava sa synchronizácia 2 inštancií, systém pracuje s primárnou inštanciou
  - V prípade zlyhania (alebo upgradu) primárnej inštancie sa DNS záznam zmení tak, aby ukazoval na sekundárnu

Čítajte viac: https://cloud.netapp.com/blog/understanding-aws-high-availability-compute-sql-and-storage

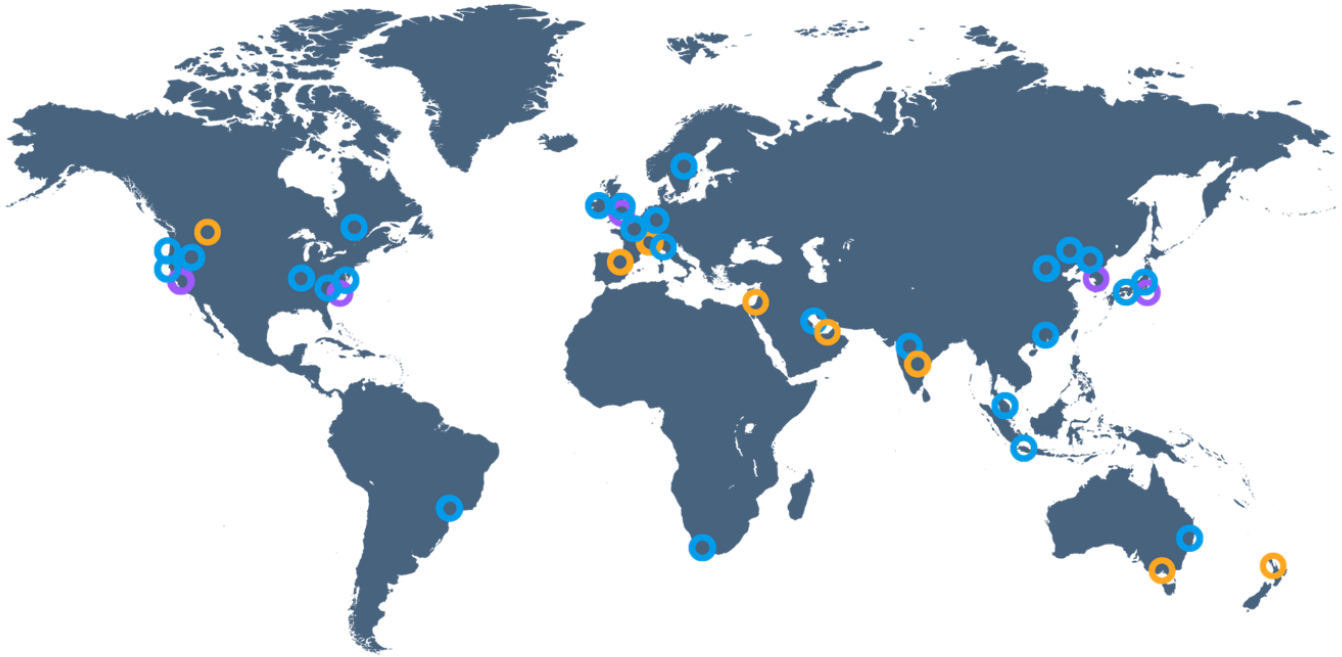
Viaczónové nasadenie a failover - príklad



Zdroj obrázka: https://medium.com/awesome-cloud/aws-difference-between-multi-az-and-read-replicas-in-amazon-rds-60fe848ef53a

Viaczónové nasadenie (multi-AZ deployment)

- Regióny (oblasti) vs. zóny dostupnosti - AZ (dátové centrá)
- AWS - 84 AZ naprieč 26 regiónmi sveta



Zdroj obrázka: What is AWS (amazon.com)

Viaczónové nasadenie (multi-AZ deployment) /2

- Viaczónové nasadenie = rozdelenie medzi viaceré dátové centrá
  - Aj v rámci určitého regiónu
  - Prevencia pred **haváriou dátového centra**
  - Uľahčenie **upgradov systému**
  - Zjednodušenie **škálovania**
  - Zvýšenie robustnosti **zálohovania**
- Optimálne je **rovnomerné rozdelenie** medzi **väčší počet dátových centier** - AZ
- Čím väčší počet uzlov (replík) v rámci dátového centra, tým väčší problém v prípade jeho zlyhania

Čítajte viac: https://aws.amazon.com/blogs/containers/amazon-ecs-availability-best-practices/

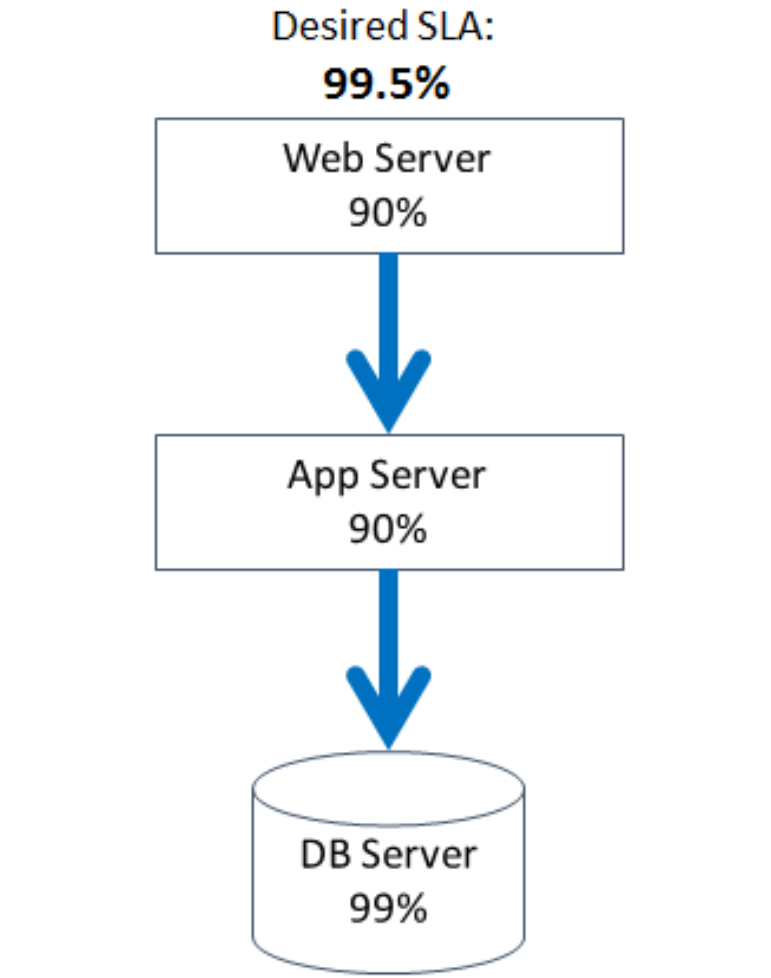
Ako dosiahnuť vysokú dostupnosť? (úvod)

- SLA (Service Level Agreement) = garantovaná úroveň služby
  - Zmluva medzi poskytovateľom a odberateľom služby
  - Merateľná prostredníctvom **metrík**, typicky dostupnosť systému v čase - **uptime** (napr. 99,9% uptime = max. 8h nedostupnosti počas 1 roka)
  - Neplnenie SLA je obvykle striktne penalizované
  - Typicky vzťah typu B2B

Ako dosiahnuť vysokú dostupnosť? (postup)

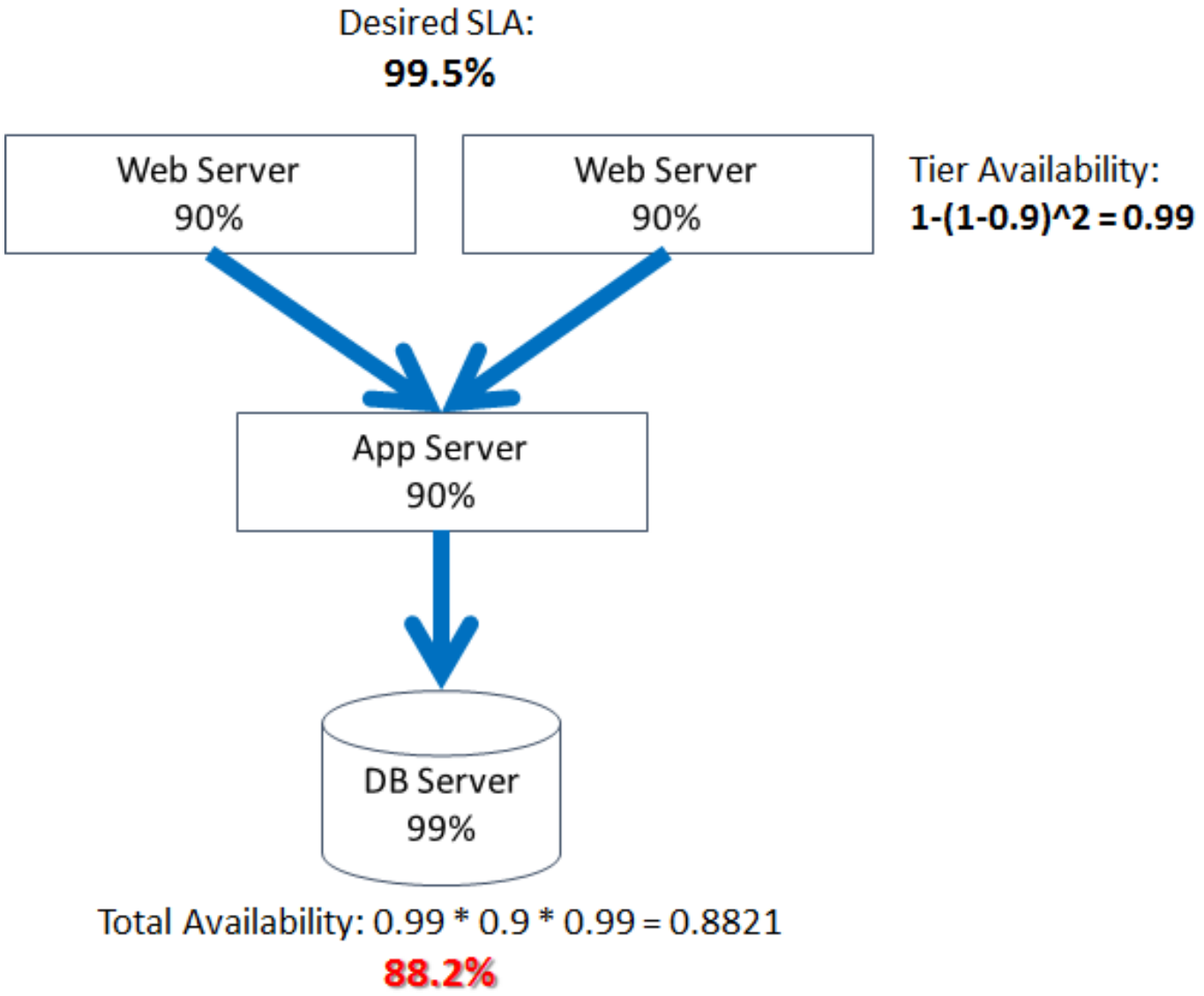
1. Rozdelenie systému na **časti (vrstvy)**
   - Výpočtová vrstva (compute)
   - Databázová vrstva (SQ)
   - Úložisková vrstva (block storage, object storage)
2. Pridanie **redundancie**
   - Najjednoduchšie pre **bezstavové komponenty** - výpočtová vrstva
   - Postupné pridávanie redundancie (replík) do ďalších vrstiev - databáza, úložisko
   - Redundancia naprieč **dátovými centrami** - zónami dostupnosti (AZ)
   - **Dostupnosť systému ako celku je súčinom dostupnosti jednotlivých vrstiev**

Ako dosiahnuť vysokú dostupnosť? (príklad)



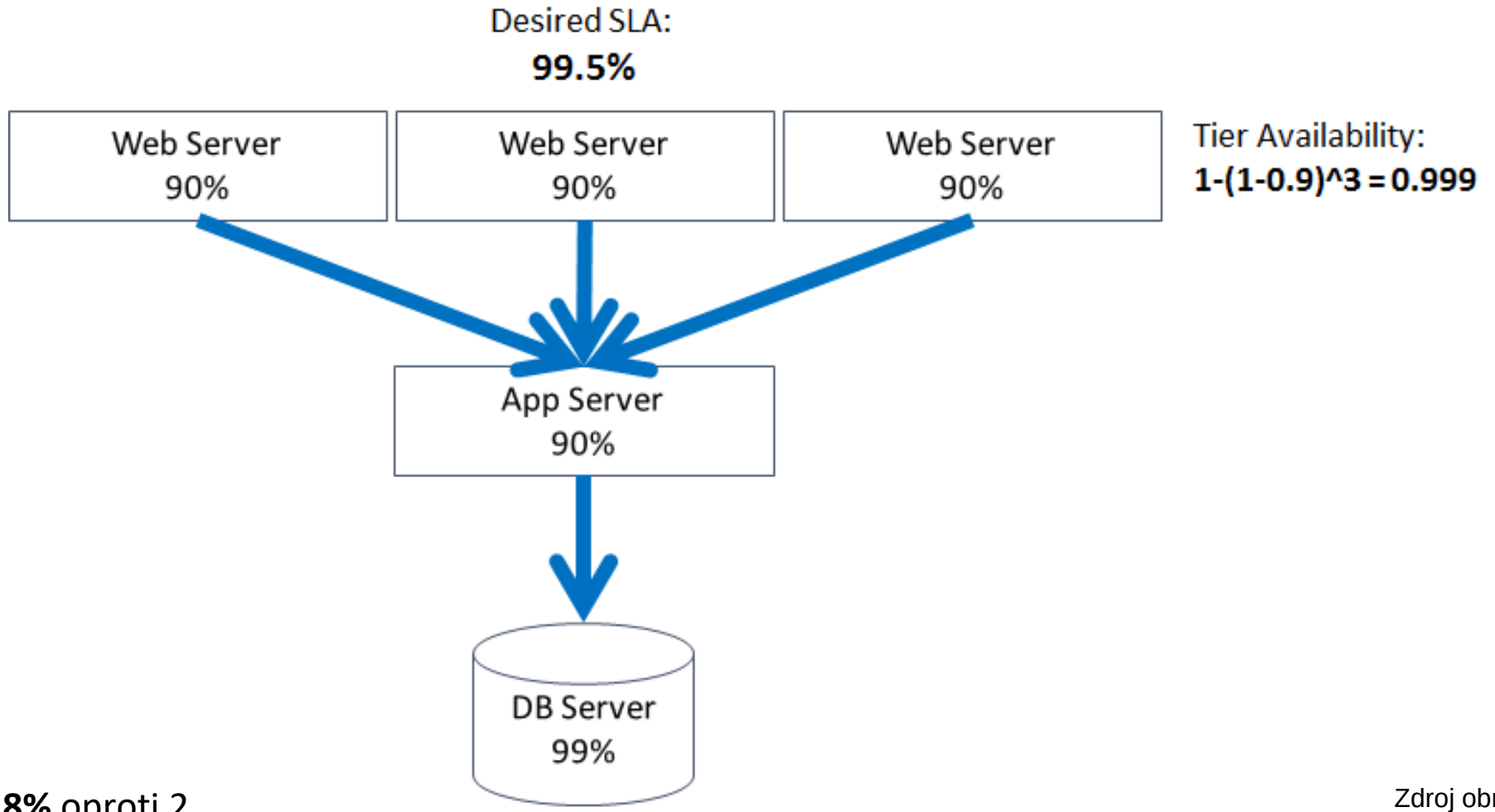
Zdroj obrázka:
https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/

Ako dosiahnuť vysokú dostupnosť? (príklad) /2



Zdroj obrázka:
https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/

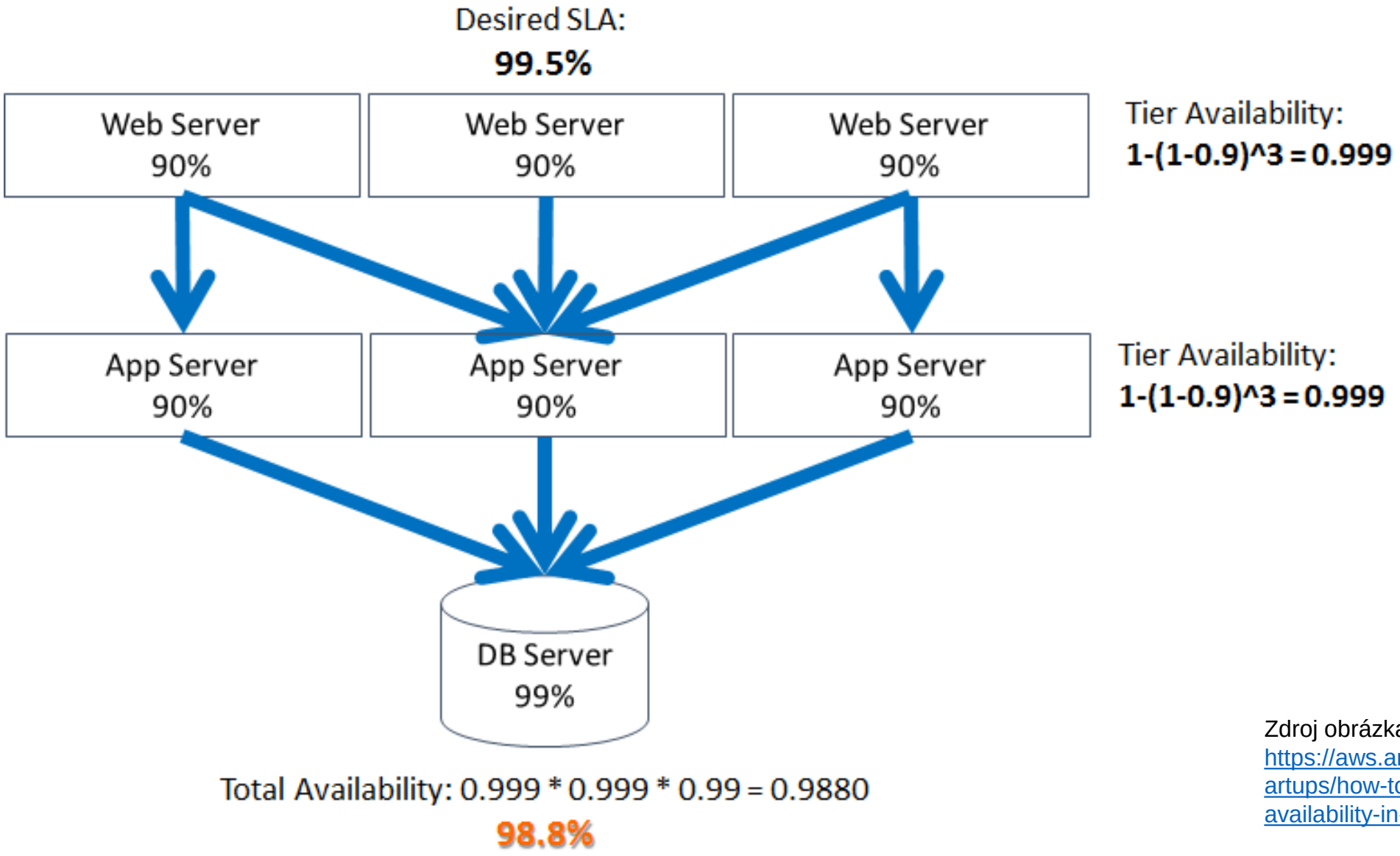
Ako dosiahnuť vysokú dostupnosť? (príklad) /3



Rozdiel iba **0,8%** oproti 2 webserverom = problém **klesajúcich výnosov** (log)

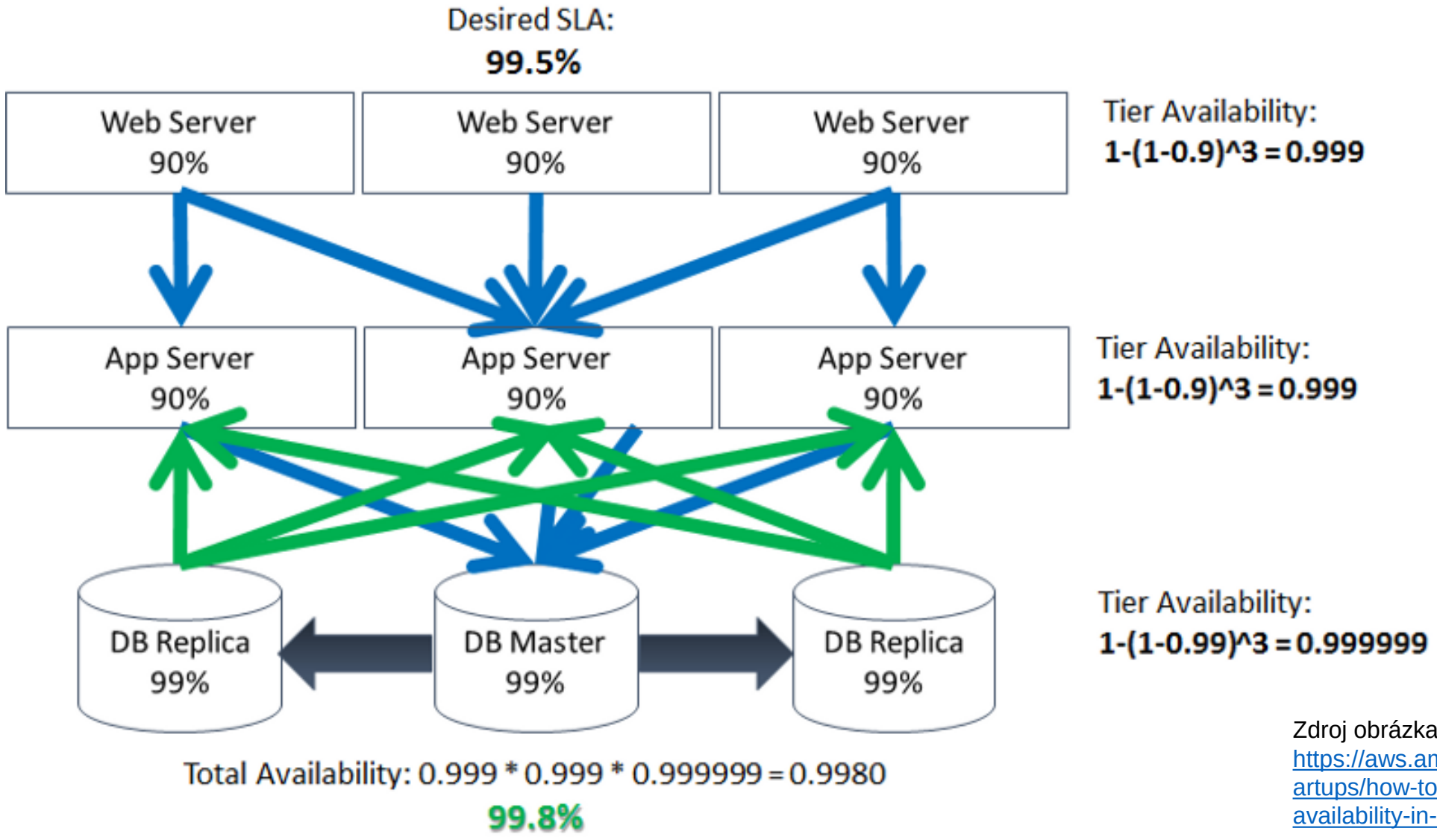
Zdroj obrázka: https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/

Ako dosiahnuť vysokú dostupnosť? (príklad) /4



Zdroj obrázka: https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/

Ako dosiahnuť vysokú dostupnosť? (príklad) /5



Zdroj obrázka: https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/

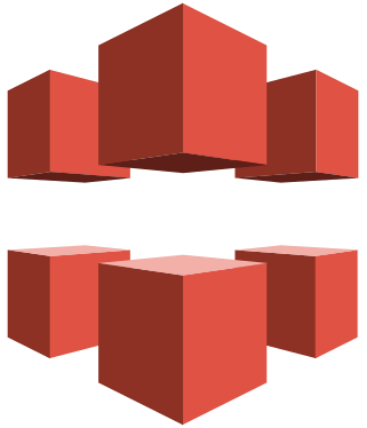
Dostupnosť služby - zhrnutie

- Rozdelenie systému na **vrstvy**
- Pridávanie **redundancie** - logaritmické zvyšovanie dostupnosti systému ako celku
  - Problém (bod) **klesajúcich výnosov** (diminishing returns)
- Pozor na dostupnosť dátového centra (definovaná v SLA)
  - **Viaczónové nasadenie** (multi-AZ deployment)
- Vysoko dostupná architektúra uľahčuje **škálovateľnosť** systému
- Dostupnosť systému ako celku < dostupnosť najslabšieho článku

Čítajte viac: https://aws.amazon.com/blogs/startups/how-to-get-high-availability-in-architecture/
https://www.weibull.com/hotwire/issue79/relbasics79.htm

Optimalizácia doručovania obsahu (CDN) - motivácia

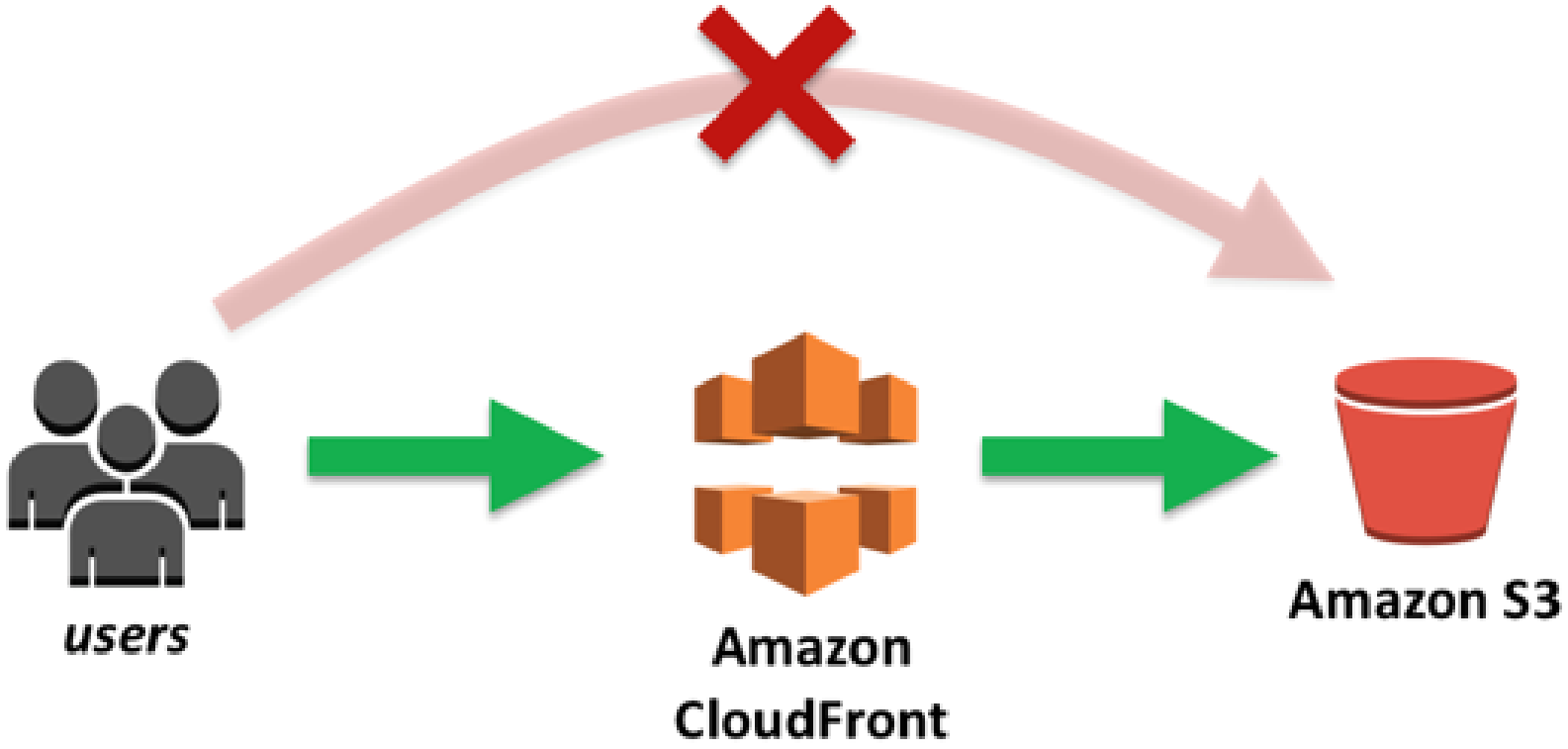
- Ako doručovať (statický) obsah klientom tak, aby bola:
  - Minimalizovaná **doba odozvy** (latency),
  - Minimalizovaná **záťaž servera** (zdroja obsahu),
  - Minimalizované **náklady** (v pomere ku kvalite služby) **a zároveň**:
  - Maximalizovaná **bezpečnosť** (najmä ochrana pred DDoS),
  - Maximalizovaná dostupná **šírka pásma** (bandwidth),
  - Maximalizovaná **efektivita** doručovania obsahu.



Optimalizácia doručovania obsahu (CDN)

- Sieť pre doručovanie obsahu (CDN = Content Delivery Network)
  - Známi poskytovatelia: Cloudflare, AWS CloudFront
- Umožňuje pristúpiť klientom k (nielen) statickému obsahu, s cieľom maximalizovať jeho **efektivitu doručovania**
  - Úložisko obsahu - väčšinou **object storage** (AWS S3), ale môže byť aj server
  - Druh obsahu - často **multimédiá** (avatary, fotky, audio nahrávky…)
- **Vyrovnávacia pamäť** pre statický obsah (cache)
  - Servery situované v rámci celého sveta (tzv. edge servers)

Optimalizácia doručovania obsahu (CDN) /2



Zdroj obrázka: https://aws.amazon.com/blogs/networking-and-content-delivery/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/

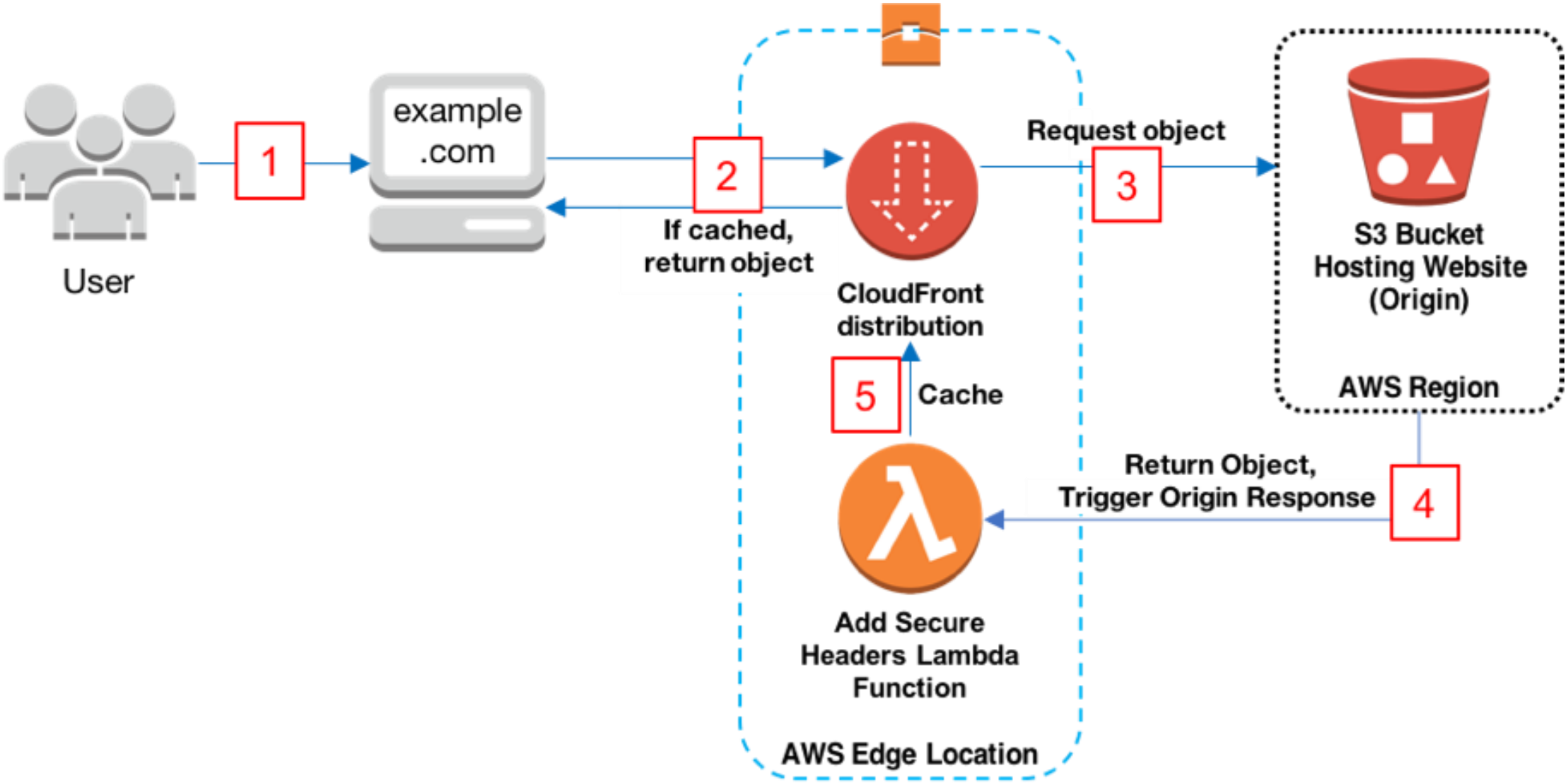
Optimalizácia doručovania obsahu (CDN) /3

- Vstupná brána pre klientov - **distribúcia záťaže**, **ochrana pred DDoS**
- Prostredník medzi klientom a obsahom - **middleware**
  - Umožňuje **dynamicky modifikovať**, príp. **generovať obsah** (AWS Lambda@Edge)
  - Napr. pridanie bezpečnostných hlavičiek (X-XSS-Protection)
  - Možné je použiť aj vlastný kód (napr. funkciu v Node.js)

Čítajte viac: https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/GettingStarted.SimpleDistribution.html

Optimalizácia doručovania obsahu (CDN) /4

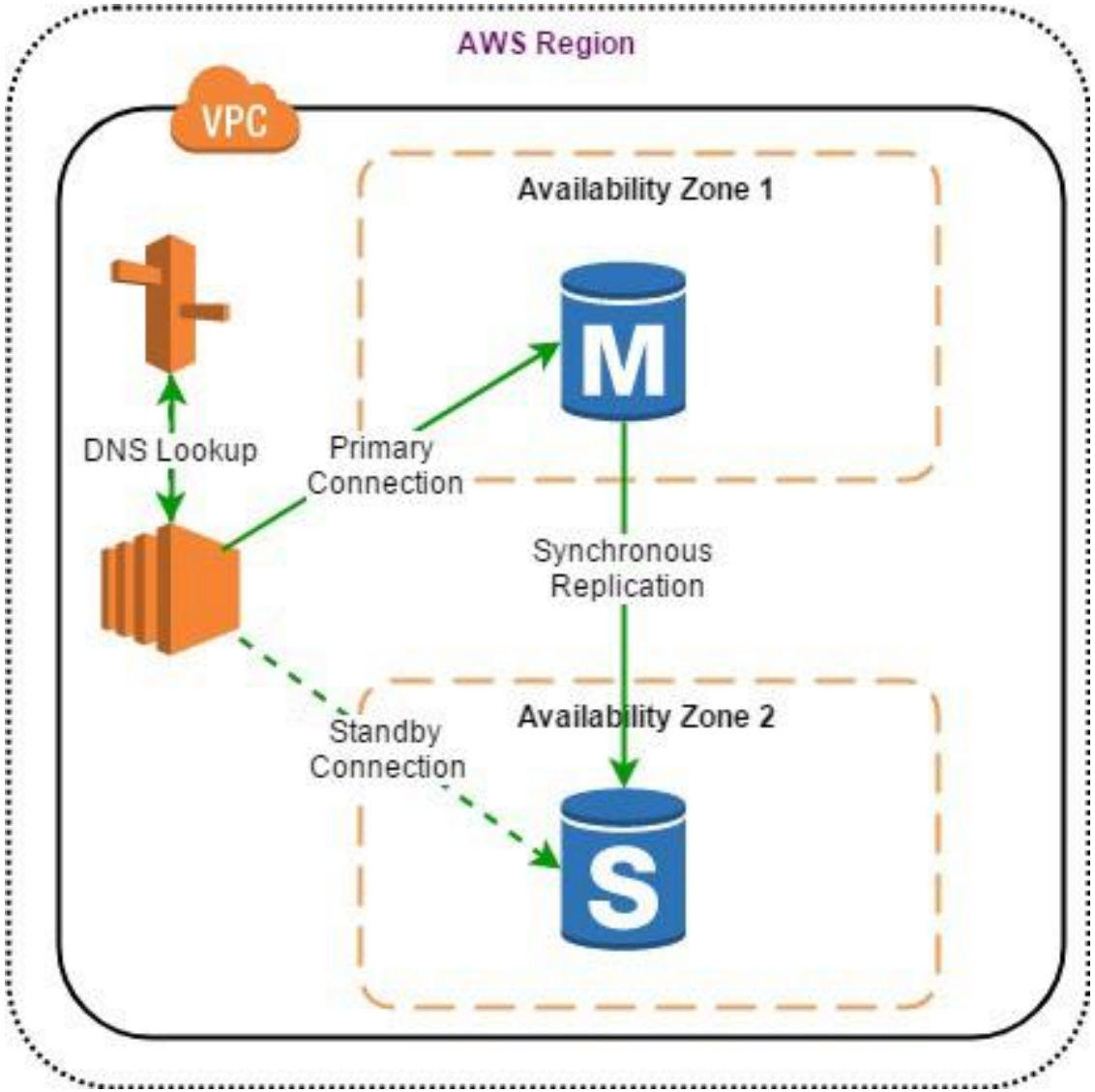


Zdroj obrázka: https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/

Replikácia a zálohovanie dát

- **Replikácia** = synchronizácia dát medzi 2 a viacerými bodmi (napr. server, databáza) v (takmer) **reálnom čase**
  - \+ Možnosť extrémne **rýchlej obnovy** - v ideálnom prípade stačí **zmeniť záznam DNS** z primárnej inštancie na sekundárnu (**failover**)
  - \+ Replikácia sa hodí aj v prípade systémových **upgradov** - **minimalizácia downtime** (failover)
  - \- Výrazne **vyššie náklady** - prevádzka “redundantnej” inštancie (inštancií), ktoré sú väčšinu času pasívne

Replikácia databázy a failover - príklad



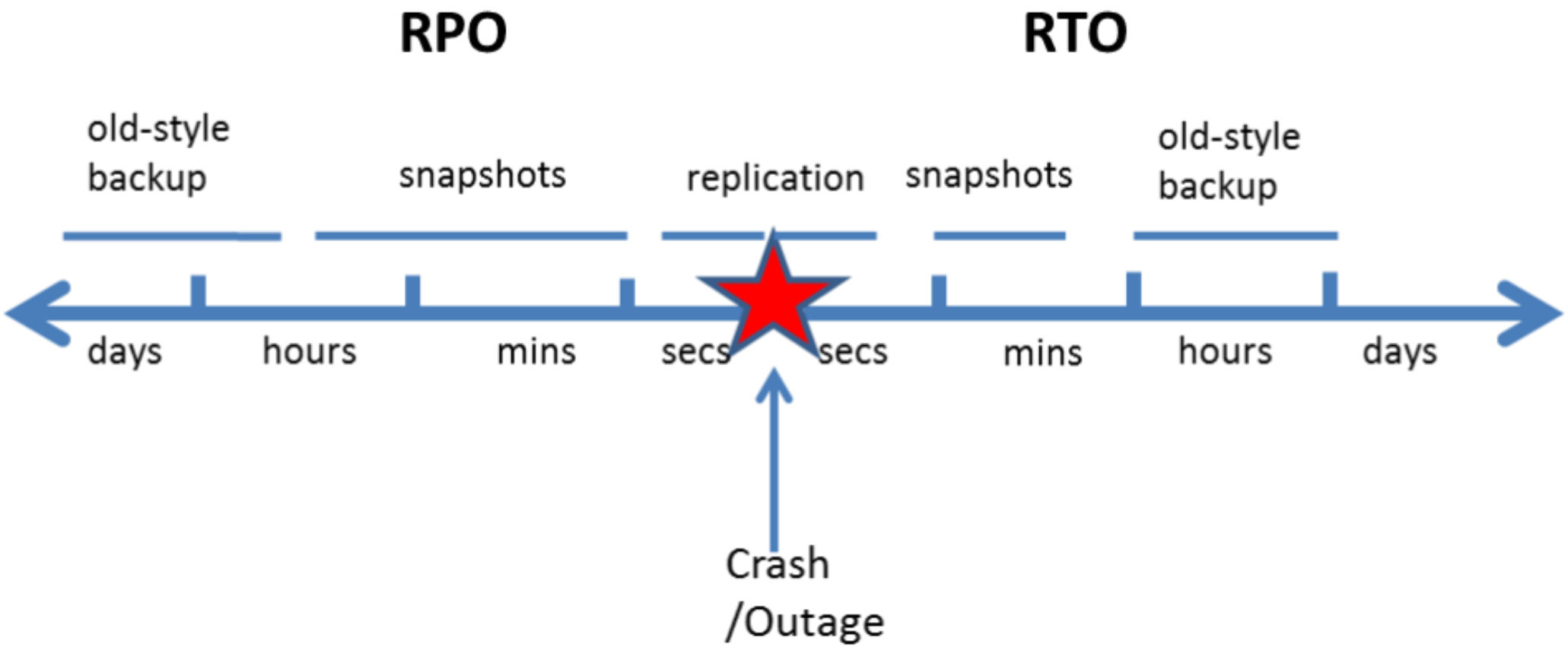
Zdroj obrázka: https://medium.com/awesome-cloud/aws-difference-between-multi-az-and-read-replicas-in-amazon-rds-60fe848ef53a

Replikácia a zálohovanie dát /2

- **Zálohovanie** = udržiavanie záložnej kópie (kópií) dát na médiu, s možnosťou ich **obnovy** v prípade havárie (potreby)
  - \+ Vyhotovenie presnej **1:1 kópie dát** v rámci **bodu v čase** (point in time)
  - \+ **Úplné** (full), **diferenciálne** (differential) a **inkrementálne** (incremental) zálohy
  - \+ Možnosť použiť rôzne druhy médií - optické (BD, AD), magnetické (páska, NAS + HDD + RAID), cloud (object storage)...
  - \- Obnova “statickej” zálohy je často komplikovanejšia a pomalšia ako v prípade použitia replikácie dát (failover) - RTO, RPO

Čítajte viac: https://www.rubrik.com/insights/difference-between-backup-and-replication
https://n2ws.com/blog/backup-and-recovery/backup-vs-repliaction-cloud-part1

Replikácia a zálohovanie dát /3



Zdroj obrázka: https://n2ws.com/blog/backup-and-recovery/backup-vs-repliaction-cloud-part1

Replikácia a zálohovanie dát - zhrnutie

- Najvhodnejší prístup je kombinácia **replikácia + zálohovanie**
- Zlaté pravidlo zálohovania (3-2-1 rule):
  - Aspoň 3 kópie
  - Na aspoň 2 druhoch médií
  - S aspoň 1 kópiou na odlišnom fyzickom mieste
- Každá záloha je lepšia ako žiadna záloha (!)

Zdroje obrázkov: https://www.4kfilme.de/wp-content/uploads/2014/03/archival-disc.jpg
https://www.backupworks.com/productimages/hp-tape/HP_LTO6_WORM.jpg

DevOps a DevOps tím

- **Dev** (Development) + **Ops** (Operations - nasadenie a prevádzka)
- Spôsob organizácie vývoja softvéru v tíme
- ‘Kombinácia **filozofie, postupov a nástrojov** umožňujúca organizácii dodávať aplikácie a služby **rýchlo a efektívne**’
- **Prelínanie funkcií** vo vývojovom tíme (programovanie, nasadzovanie)
- Previazanosť s vlastnosťami mikroslužieb (kohéznosť, zapuzdrenosť a i.) a cloudu (agilita, elasticita)

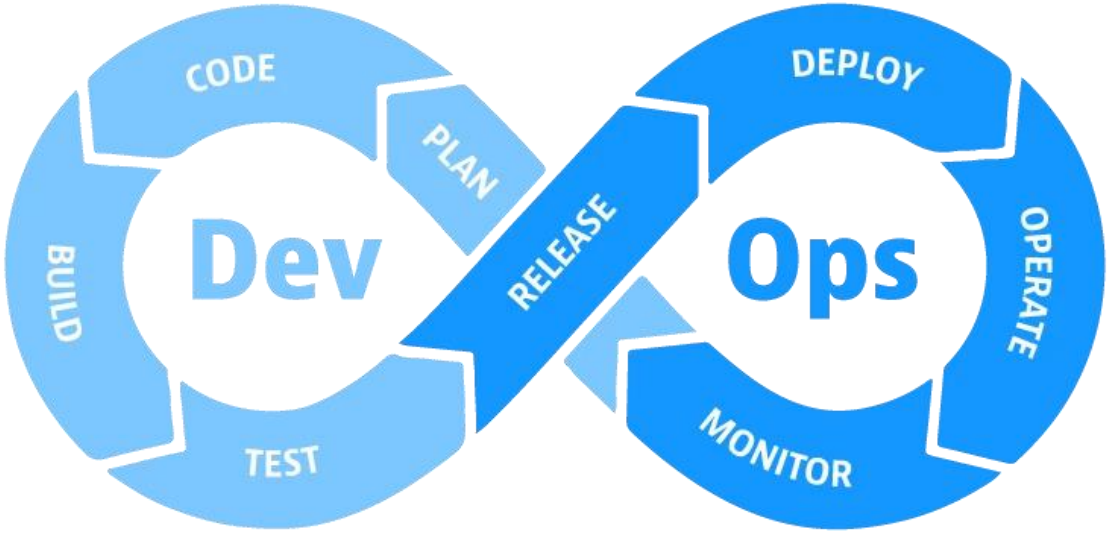
Zdroj citácie: What is DevOps? - Amazon Web Services (AWS)

DevOps tím - výhody

+ Rýchlosť - rapídne dodávanie funkcionality do produkcie, reagovanie na meniaci sa trh (konkurenciu)
+ Kvalita - techniky ako **testami riadený vývoj (TDD)**, **kontinuálna integrácia a nasadenie (CI/CD)** prispievajú k celkovej kvalite softvéru
+ Komunikácia a kolaborácia - vývojári sa spolupodieľajú na nasadzovaní a prevádzke a naopak
  - lepšie pochopenie sa, úspora času, efektivita

DevOps tím - nevýhody

- Zle fungujúci DevOps = chaos - plná adherencia k DevOps filozofii môže byť v praxi ťažšie dosiahnuteľná



Čítajte viac: [What is DevOps? - Amazon Web Services (AWS)]

Kontinuálna integrácia a nasadenie (CI/CD)

- Metodológia vývoja softvéru, umožňujúca zdrojový kód **priebežne** a **automatizovane testovať**, **integrovať** a **nasadzovať**
- Výhody CI/CD:
  - Rýchlejšie dodávanie **inkrementálnych** prírastkov funkcionality do produkcie
  - Rýchlejšie **nájdenie chýb** (automatizované testy)
  - Kratšie **vývojové cykly** (menší celok funkcionality)
  - Zvýšenie **produktivity** a **efektivity** vývoja (automatizácia buildu a nasadenia)

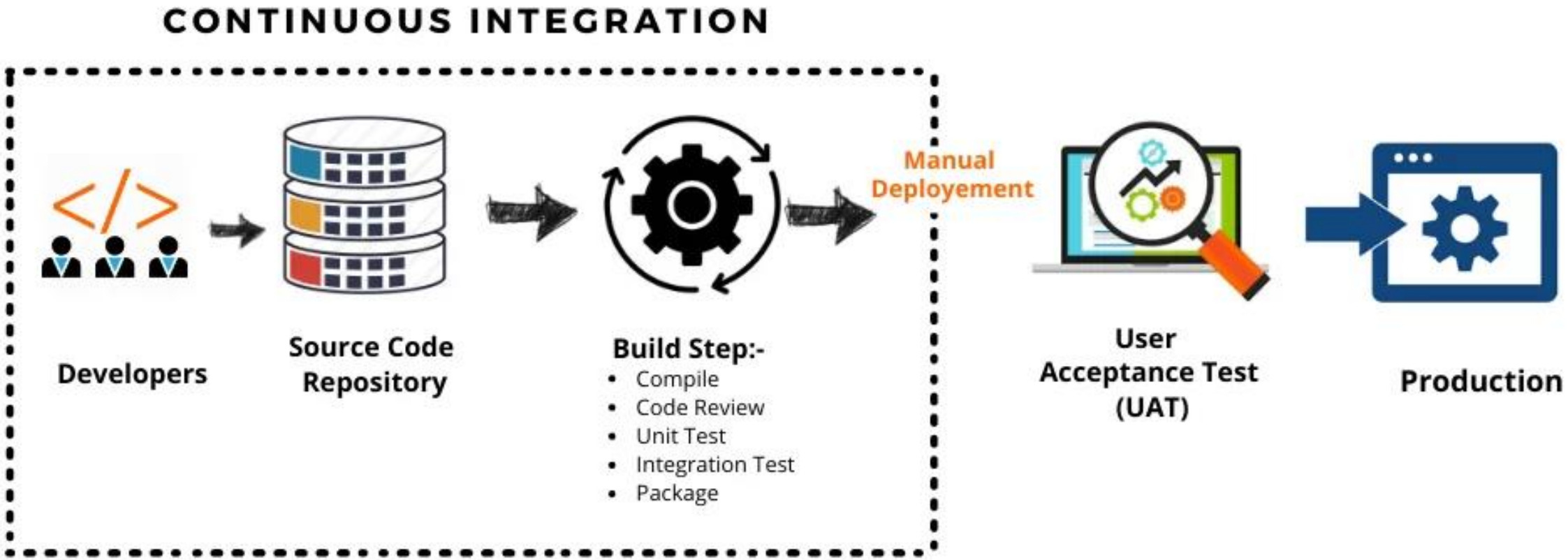


Čítajte viac: https://aws.amazon.com/devops/continuous-integration/

Kontinuálna integrácia (CI = Continuous Integration)

1. Vývojár implementuje funkcionalitu a overí jej základnú funkčnosť
    a. Lokálne prostredie (localhost) - **základné otestovanie** (manuálne, automatické)
2. Vývojár zdrojový kód odovzdá (commit) do repozitára (Git)
    a. Sada **automatizovaných testov** (**zostavenie** - build, **jednotkové** - unit, **integračné** - integration)
    b. **Zlúčenie** (merge) s hlavnou vývojovou vetvou (dev / develop)

Kontinuálna integrácia (CI = Continuous Integration) /2



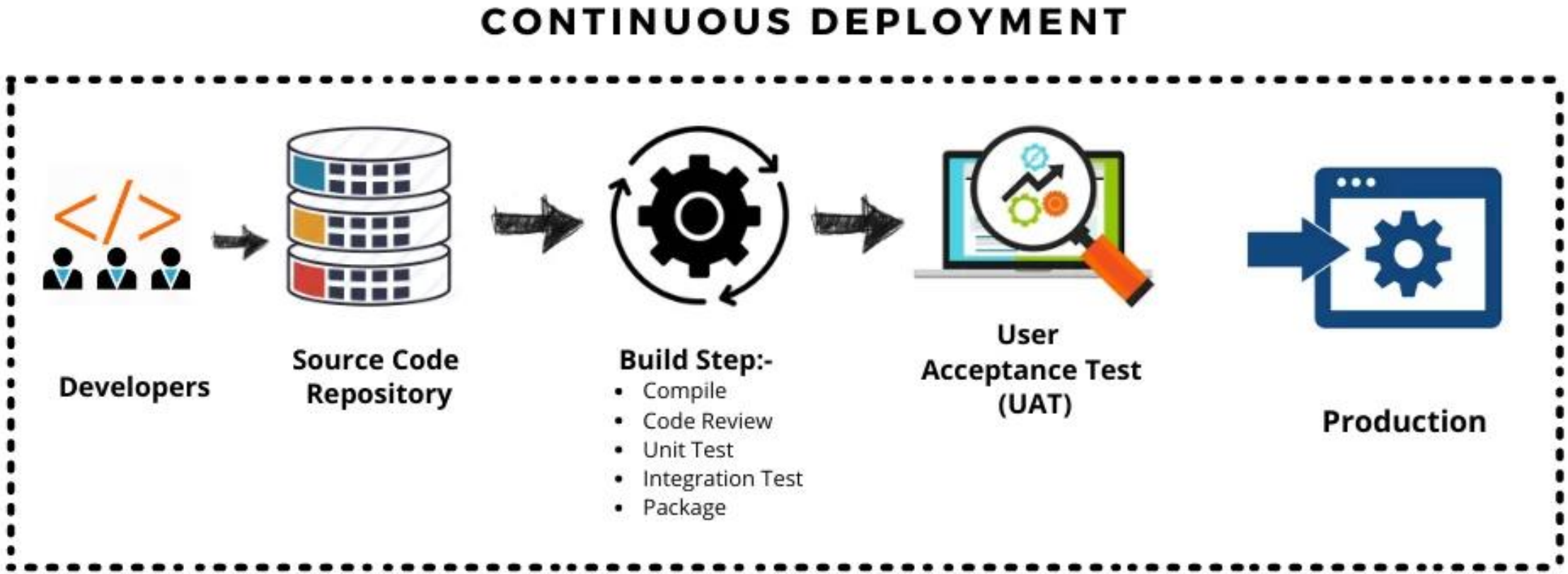
Zdroj obrázka: https://dwops.com/blog/difference-between-continuous-integration-continuous-delivery-and-continuous-deployment/

Kontinuálne nasadenie (CD = Continuous Deployment)

3. Nasadenie na **predprodukčné prostredie** (continuous delivery)
   a. **Automatizované** nasadenie na predprodukčné prostredie (development, staging)
   b. Dodatočné testovanie (automatizované aj manuálne) - **výkonnostné** (load), **zážažové** (stress), **bezpečnostné** (security, penetration)
4. Nasadenie na **produkčné prostredie** (continuous deployment)
   a. **Automatizované** nasadenie na produkčné prostredie (production)
   b. Môže sa využiť A/B nasadenie (**blue-green deployment**) - nová verzia je pripravená na okamžité “vypustenie” do produkcie (prepne sa záznam DNS)

Čítajte viac: https://aws.amazon.com/devops/continuous-integration/
https://aws.amazon.com/devops/continuous-delivery/

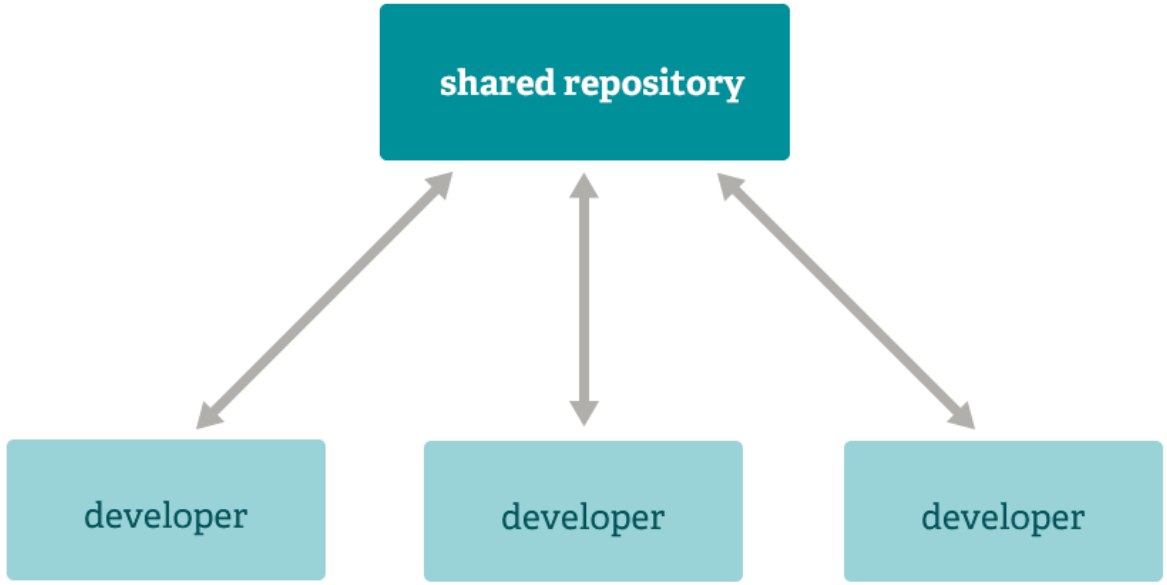
Kontinuálna nasadenie (CD = Continuous Deployment) /2



Zdroj obrázka: https://dwops.com/blog/difference-between-continuous-integration-continuous-delivery-and-continuous-deployment/

Správa verzií - version control (Git)

- Distribuovaný systém pre správu verzií (version control)
- Ukladanie, organizácia a udržiavanie **celej histórie zmien kódu**
  - Kópia (záloha) na serveri a u **každého vývojára** s naklonovaným repozitárom
- Otvorený (open-source) de facto štandard



Zdroj obrázka: https://git-scm.com/about/distributed

Správa verzií - version control (Git) /2

- Systém **vetiev (branches)** - minimálne odporúčané využitie:
  - **Master (main)** - Hotový, plne otestovaný, zintegrovaný, funkčný kód; kedykoľvek nasaditeľný do **produkčného prostredia**
  - **Dev (develop)** - Takmer hotový, funkčný kód; otestovaný aspoň základnou sadou testov; nasaditeľný do **preprodukčného prostredia**
  - Feature/<názov_funkcionality> - Pracovný kód, slúži ako distribuované úložisko pre vývojára (vývojárov), commity sú **priebežné**, **atomické** a **časté**; poskytuje informáciu o **stave vývoja** funkcionality

Správa verzií - version control (Git) /3

- Pri commitovaní je vysoko odporúčané dodržiavať **konvenciu**:
  - <commit_type>(<optional_scope>): <commit_msg_content>
- Príklady:
  - feat(app/Models): add database model for Book entity ✅
  - fix(providers/src/Library): fix issue with (...) when (...) is performed ✅
  - chore: bump Puppeteer version to 2.1.6 ✅
  - implement functionality XYZ ❌

Čítajte viac:
https://www.conventionalcommits.org/

Automatizácia nasadenia (Jenkins)

- Otvorený softvér pre tvorbu automatizovaných **DevOps pipelines**:
  - Zostavenie aplikácie (build)
  - Automatizované testovanie aplikácie (test)
  - Nasadenie aplikácie (deploy)
- Pipeline = sekvencia krokov, pozostáva z **príkazov (skriptov)**
- Jednoduchá integrácia **Jenkins + Docker**
- Široká rozšíriteľnosť prostredníctvom **zásuvných modulov** (plugins)

Jenkins

Jenkins build & deploy pipeline (Jenkinsfile) - príklad

```
node {
    stage('Clean workspace') {
        cleanWs()
    }
    stage('Preparation') {
        git branch: 'master', credentialsId: 'bitbucket', url: 'git@bitbucket.org:MyDevTeam/slek-server.git'
    }
    stage('Build image') {
        sh "docker buildx build --platform=linux/arm64 --tag=slek-server --build-arg API_URL=https://api.slek.com --load ."
        sh "docker tag slek-server:latest 12345678.dkr.ecr.eu-north-1.amazonaws.com/slek-server:latest"
    }
    stage('Push image') {
        def LOGIN_CMD = sh label: 'Getting login credentials', script: 'aws ecr get-login --region eu-north-1'
        sh label: 'AWS login', script: LOGIN_CMD
        sh label: 'Pushing image', script: "docker push 12345678.dkr.ecr.eu-north-1.amazonaws.com/slek-server:latest"
    }
    stage('Deploy image') {
        sh label: 'Deploying', script: "aws ecs update-service --cluster slek-cluster --service slek-server --force-new-deployment"
    }
}
```

Testovanie aplikácií

- Ako overiť funkcionalitu softvéru predtým, ako bude dodaný klientovi (do produkcie)?
- **Manuálne** - vytvorenie “náhodných” dát, preklikanie si implementovanej funkcionality, letmá kontrola výstupu, vizuálu…
- Problémy:
  - Ak to urobí vývojár, môže byť “zaslepený” (biased)
  - Ak to urobí tester (QA pracovník), nemusí plne chápať dosahy zmien v kóde
  - Navyše, je to časovo náročná a často “otravná” aktivita…

Testovanie aplikácií /2

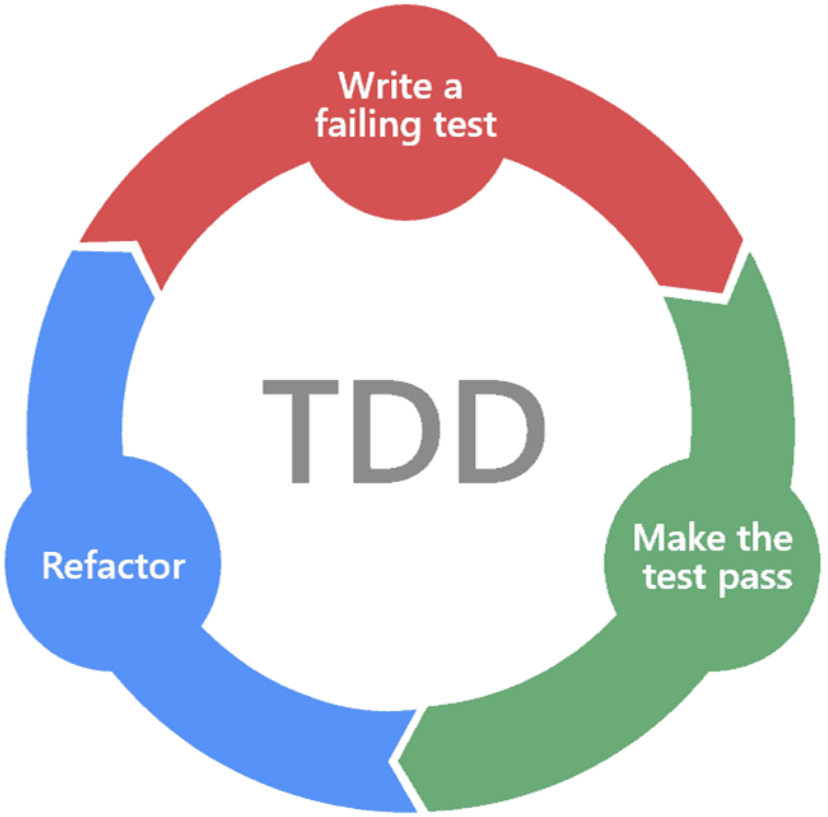
- Druhá možnosť - **automatizované testy**
- Dobrý test by mal:
  - **Zvyšovať kvalitu** - produktu aj kódu,
  - **Znižovať riziko** = **náklady** spojené s chybami (**regresie** - keď nové zmeny v kóde pokazia existujúcu funkcionalitu),
  - **Zlepšovať porozumenie** - kódu aj softvéru ako celku,
  - Byť **jednoducho napísateľný** a **spustiteľný** - test by nemal byť nepriateľom,
  - Byť **ľahko udržiavateľný** - aktualizácia kódu by nemala znamenať, že treba prepísať existujúce testy.

Čítajte viac: [The 6 Principles of Test Automation - Semaphore (semaphoreci.com)]

Test Driven Development (TDD)

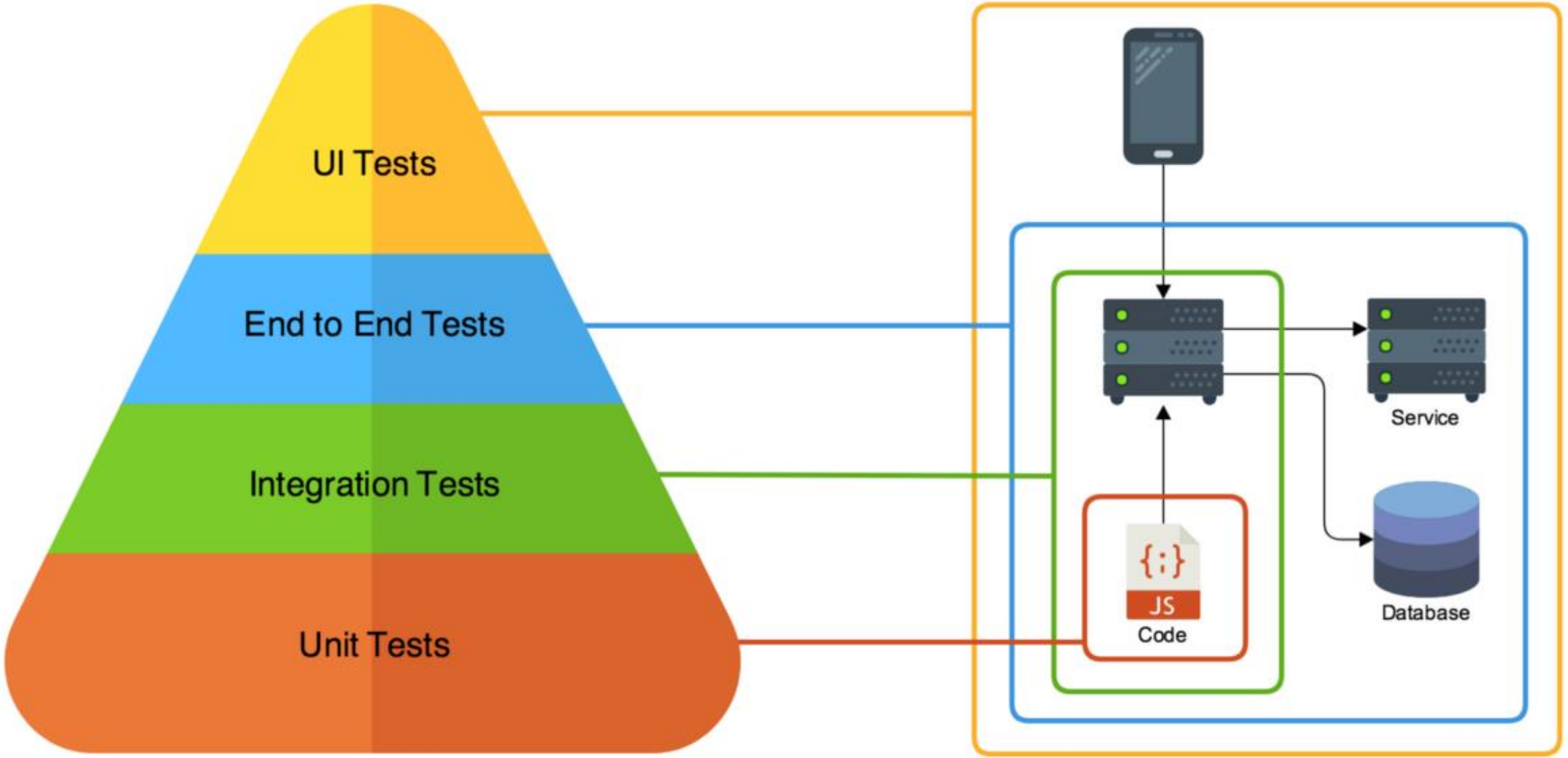
- Testami riadený vývoj = metodológia vývoja softvéru, kde **implementácii** každej funkcionality **predchádza napísanie testu** k nej

1. Výber funkcionality na implementáciu
2. Napísanie testu (testov)
3. Napísania iba takého kódu, ktorý vedie k prejdeniu testu
4. Refaktoring - vyčistenie kódu
5. Pokračovanie krokom 1



Čítajte viac: Test-driven development - IBM Garage Practices
Zdroj obrázka: Why Test-Driven Development (TDD) | Marsner Technologies

Funkcionálne testovanie - hierarchia



Zdroj obrázka: [Automated Testing: The Cornerstone of CI/CD - Semaphore (semaphoreci.com)]

Jednotkový test (unit test)

- Je **atomický** - preveruje funkcionalitu konkrétnej, malej časti kódu
- Je **izolovaný** - nevyžaduje externé závislosti (databázu, mikroslužby…)
  - Využívajú sa **náhrady** (test doubles / mocks / stubs) - vygenerované **“statické” dáta** a **pseudo funkcie**, ktoré nahrádzajú externú závislosť
- Syntax **Arrange-Act-Assert** (AAA):
  - **Arrange** - Príprava testovacieho prostredia (dáta, stav objektov…)
  - **Act** - Zavolanie testovanej funkcionality (metódy, ktorá vracia výsledok)
  - **Assert** - Validácia, či krok Act skončil požadovaným výsledkom (napr. návratová hodnota metódy).
- Populárne rámce pre Node.js - Mocha, Jest, Jasmine, Japa

Čítajte viac: Node.js Unit Testing: Get Started Quickly With Examples (testim.io)

Jednotkový test (unit test) - príklad (AdonisJS - Japa)

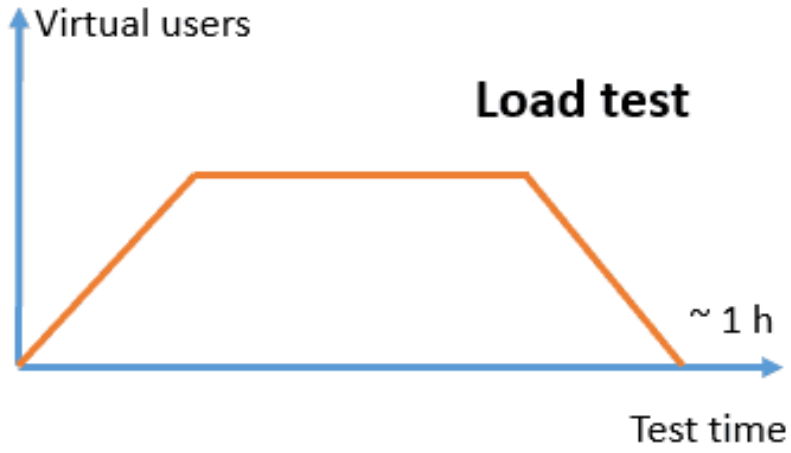
```
import { test } from '@japa/runner'

test('display welcome page', async ({ client }) => {
  //arrange
  const route = '/'

  //act
  const response = await client.get(route)

  //assert
  response.assertStatus(200)
  response.assertTextIncludes('<h1 class="title"> It Works! </h1>')
})
```

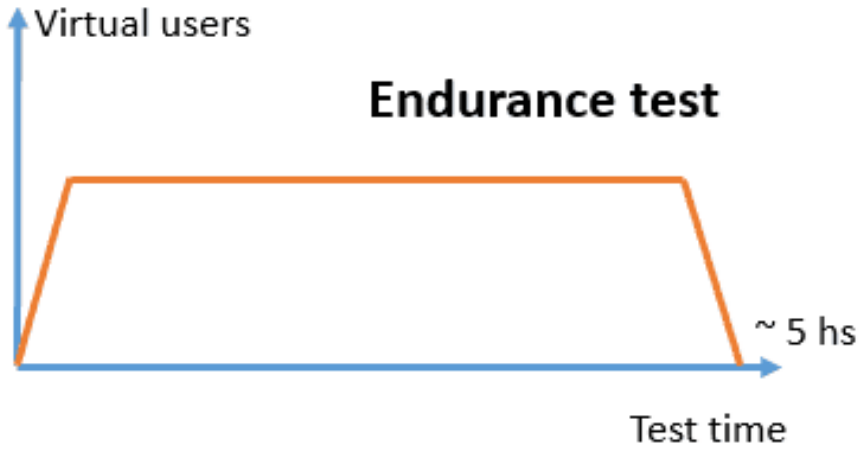
Viac o AdonisJS testovaní:
Introduction (adonisjs.com)

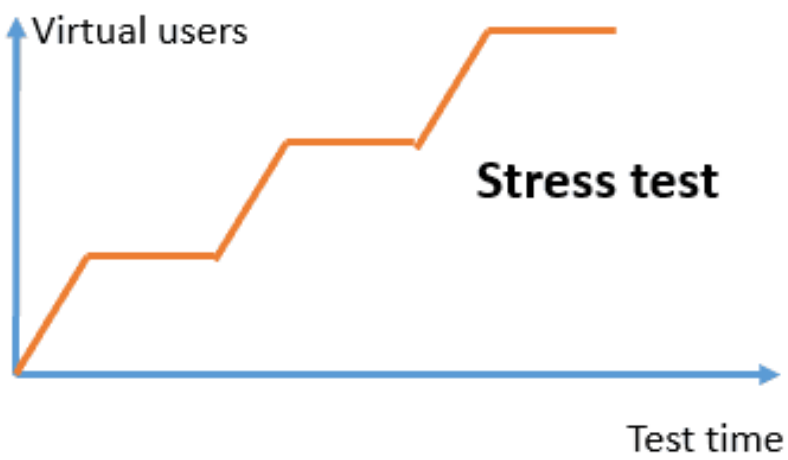
Ďalšie druhy testov - nefunkcionálne testovanie

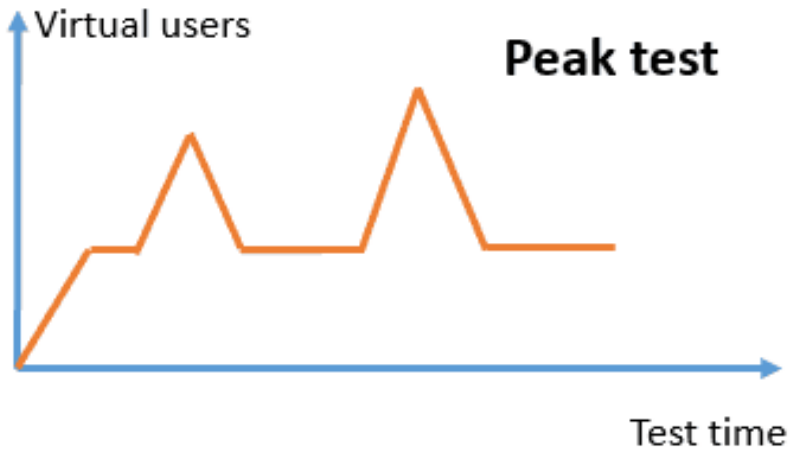
- Výkonnostné testovanie (load testing)
  - Testovanie systému pri **typickej záťaži** (napr. pracovný deň)
- Záťažové testovanie (stress testing)
  - Determinovanie absolútnych **výkonnostných limitov** systému
- Špičkové testovanie (spike / peak testing)
  - Ako systém zvládne **krátkodobú extrémnu záťaž**
- Vytrvalostné testovanie (endurance / soak testing)
  - Ako je systém schopný vydržať určitú záťaž **dlhodobo**

Nefunkcionálne testovanie - porovnanie









Zdroj obrázka: [All About Endurance Testing With TestOS | TestGrid | Blog]

Bezpečnosť a penetračné testovanie

- Penetračné testovanie = **aktívny**, **cieľavedomý** a **systematický** prístup k testovaniu a vyhodnoteniu **bezpečnostných vlastností systému**
- Metóda **bielej skrinky** (white box)
  - Vychádzame z **apriórnych znalostí o systéme** - komponenty, technológie, kód
  - Nižšia časová náročnosť, problémom môže byť “zaslepenie” (bias)
- Metóda **čiernej skrinky** (black box)
  - Nemáme informácie o systéme, kľúčový je zber informácií (reconnaisance)
  - Náročnejšie na realizáciu, ale výpovednejšie

Čítajte viac: HUDEC, L. Penetračné testovanie. In Bezpečnosť v Internete [online]. 2021. Dostupné na internete: http://www2.fiit.stuba.sk/~lhudec/SIN/Penetracne_testovanie.ppt

Penetračné testovanie - postup

1. Pasívny zber informácií a prieskum systému (**reconnaisance**)
   a. Zoznam otvorených portov, verzia OS, serverov (prístupná status stránka)...
   b. Nástroje Nmap, Recon-ng, Nikto, Dirb
2. Aktívne skenovanie a mapovanie zraniteľností (**scanning**)
   a. Vystavené citlivé údaje (napr. konfigurácia), slabá autentifikácia (neobmedzený počet pokusov pre zadanie hesla - útok hrubou silou)
   b. Metasploit, Nmap, Nikto

Penetračné testovanie - Nikto (príklad)

```
kalikali@kali:~$ nikto -h 192.168.229.137
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.229.137
+ Target Hostname:    192.168.229.137
+ Target Port:        80
+ Start Time:         2020-04-14 10:59:53 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent
+ The X-Content-Type-Options header is not set. This could allow the user agent to r
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.37). Apache 2.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easil
ives for 'index' were found: index.php
+ Web Server returns a valid response with junk HTTP methods, this may cause false p
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ /phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3268: /doc/: Directory indexing found.
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially s
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially s
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially s
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially s
+ OSVDB-3092: /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL databases,
+ Server may leak inodes via ETags, header found with file /phpMyAdmin/ChangeLog, in
+ OSVDB-3092: /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting ...
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo()
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpMyAdmin/: phpMyAdmin directory found
```

Penetračné testovanie - postup /3

3. Útočenie a zneužitie zraniteľností (**exploitation**)
   a. SQL injekcie (injections), medzistránkové skriptovanie (XSS), útok hrubou silou (brute force - napr. slovníková metóda)
   b. Metasploit, Sqlmap, Hashcat, Burp suite, Nmap (skripty)
4. Vyhodnotenie zraniteľností (**post-exploitation**)
   a. Identifikácia, kvantifikácia (ohodnotenie), mapovanie na útoky, náprava
   b. Burp suite - umožňuje vygenerovať automatizovanú správu o bezpečnosti

Čítajte viac: 5 Important Questions to Ask Your Next Penetration Tester (redcanari.com)

OWASP (Open Web Application Security Project)

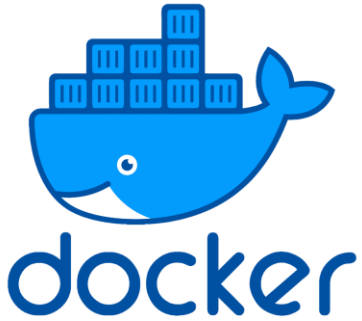
- **OWASP Top 10** - 10 najkritickejších zraniteľností web aplikácií:
  - **Injekcie** - odoslanie škodlivých dát interpreteru (SQL injection), vykoná neoprávnený príkaz (dopyt)

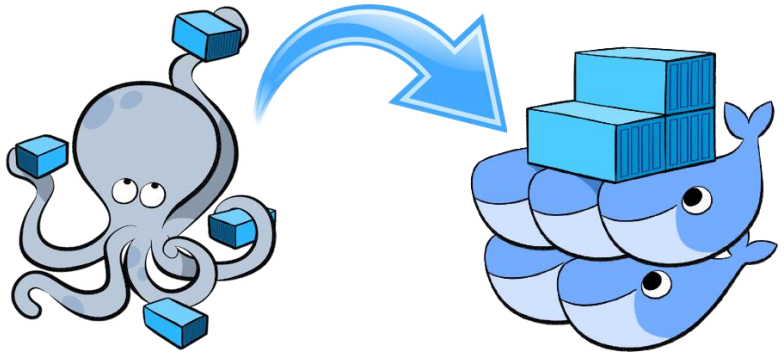
    => validácia dát, ORM
  - **Chybná autentifikácia** - povolenie slabých hesiel, neobmedzený počet pokusov…
  - **Vystavenie citlivých dát** - napr. konfigurácia webservera (Apache)
  - Ďalšie: zlá bezpečnostná konfigurácia, XSS, XEE, nedostatočné logovanie, nebezpečná deserializácia, používanie zraniteľných komponentov

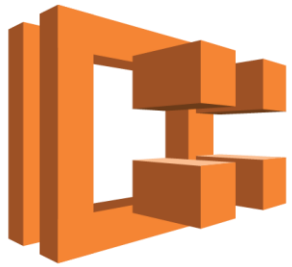
Čítajte viac: [OWASP Top Ten Web Application Security Risks | OWASP]

Zhrnutie DevOps časti na predmete VPWA

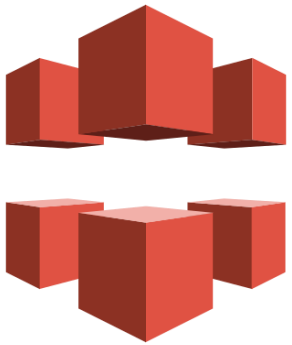
- Mikroslužby, SW kontajnery
- Orchestrácia kontajnerov
- Cloud computing
- Dostupnosť služby, CDN, replikácia
- DevOps, CI/CD, automatizácia
- Testovanie aplikácií, TDD







AWS ECS



- Spätná väzba??? ☺