DOSTUPNOSŤ A ŠKÁLOVANIE INTERNETOVÝCH APLIKÁCIÍ V CLOUDE

SPRIEVODCA TVORBOU SKUTOČNÝCH CLOUDOVÝCH APLIKÁCIÍ

ZAŽIME TO SPOLU

OBSAH

Éra cloudu **3**

Dve Architektúry webov **3**

Monolitické Aplikácie 4

Cloud ready aplikácie 4

Domáce zvieratá a dobytok **5**

Cloud a virtualizácia **5**

Infraštruktúra ako kód **6**

Agilná IT infraštruktúra 6

Automatizácia 7

Testovacie prostredia 7

Typická webová architektúra 8

Skupiny serverov 8

Prezentačná vrstva (bezstavová) 8

Aplikačná vrstva (bezstavová) 9

Backend vrstva (stavová) 9

Záver **10**

ÉRA CLOUDU

Éra cloudu je tu, niet o tom pochýb. Hlavné dôvody na vstup do cloudu, flexibilita a nižšie náklady, sa stávajú neodolateľnými, ale do popredia sa dostáva aj s cloudom súvisiaca dynamika a bezpečnosť. Preto by mala byť cloudová stratégia súčasťou IT plánovania v každej organizácii, bez rozdielu veľkosti či oblasti podnikania.

Internetové aplikácie boli medzi historicky prvými, ktoré firmy či inštitúcie do cloudu presúvali. Vyžadujú si totiž silné a stabilné pripojenie na internet, čo je klasická doména servisných providerov. Weby navyše potrebujú flexibilnú infraštruktúru a robustnú sieťovú ochranu. A tu má Telekom nepopierateľnú výhodu lídra telekomunikačného trhu, preto už roky hostuje mnoho významných slovenských webových portálov či e-shopov.

Tento dokument sa zaoberá technickými špecifikami prevádzky veľkých a vysokodostupných internetových aplikácií. Ponúka riešenie a návrh architektúry pre horizontálne škálované weby, ktoré na platforme TelekomCloud vedia uspokojiť aj tie najnáročnejšie prevádzkové požiadavky. Zaoberá sa aj „best-practice“ postupmi na automatizáciu správy, testovanie a vývoj webov.

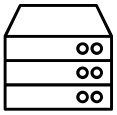
DVE ARCHITEKTÚRY WEBOV

Bežné weby sú prevádzkované na jednom serveri. Najekonomickejšia platforma v podobe štandardných webhostingových služieb stačí bežným webovým aplikáciám a malým až stredne veľkým e-shopom.

Online aplikácie, ktoré potrebujú silnejšie zdroje, sú typicky prevádzkované na vlastných virtuálnych či dedikovaných serveroch.

Pokiaľ však začne webová aplikácia narážať na limity jedného servera a je nutné, aby dosahovala vyšší výkon, je potrebné uvažovať, akým spôsobom bude prevádzkovaná na viacerých serveroch. Aby dokázala aplikácia zvládnuť nápor väčšieho množstva používateľov, musí vedieť efektívne využiť zdroje na viacerých virtuálnych serveroch.

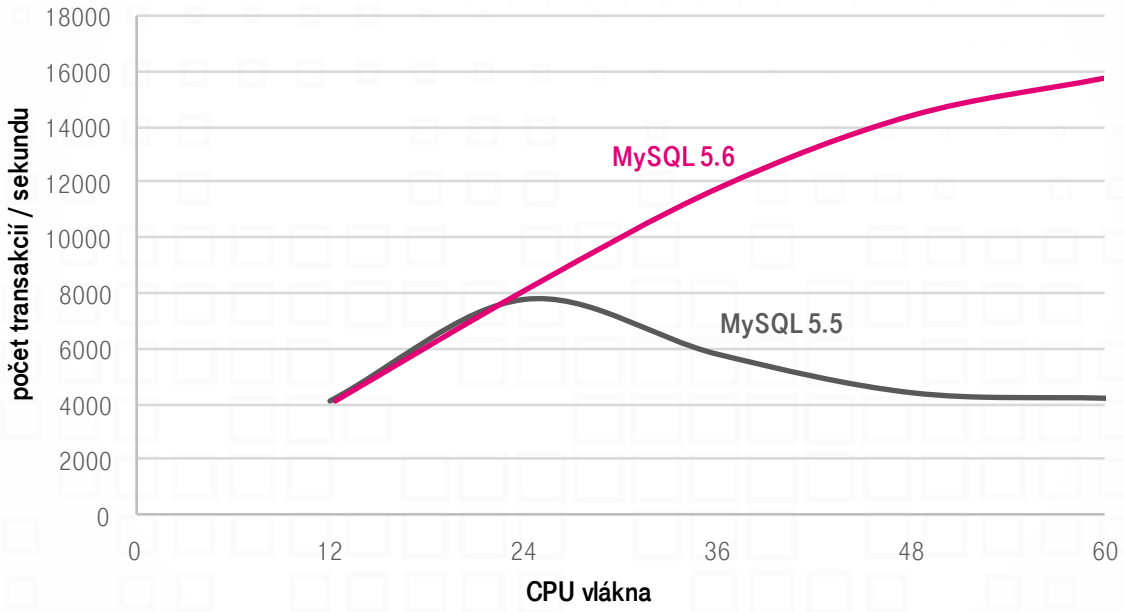
Monolitické aplikácie



Cloud ready aplikácie

MONOLITICKÉ APLIKÁCIE

Ide o staršie a stále dominantné aplikácie, ktoré boli navrhované na prevádzku na jednom serveri. V prípade vyšších nárokov na výkon sa ráta s navyšovaním zdrojov samotného servera, t. j. s vertikálnym škálovaním. Problém tohto prístupu spočíva v tom, že výkon aplikácie nemusí rásť lineárne s množstvom zdrojov, ktoré na svoj beh dostane. Niekedy môže byť tento nárast dokonca záporný, ako ukazuje graf nižšie, kde vidíme **závislosť počtu databázových transakcií od počtu CPU**. Narastajúce množstvo zdrojov napríklad pri staršej verzii MySQL servera začne odhaľovať interne úzke hrdlá v architektúre aplikácie, a preto napriek zvýšeným zdrojom dochádza k zníženiu výkonu.



Dosiahnuť vysokú dostupnosť aplikácie je ešte náročnejšie ako ju škálovať, lebo aplikácia na to nie je typicky nadizajnovaná. Preto mnoho súvisiacej komplexnosti musí byť presunutej do cloudovej infraštruktúry, ktorá však, naopak, nevie nič o aplikácii, ktorú prevádzkuje.

Do tejto kategórie patria bežné databázové servery a rôzne vnútropodnikové aplikácie. Na tieto aplikácie sa špecializuje VMware, používaný aj na platforme TelekomCloud. VMware prevádzkuje aplikácie tak, aby zakryl detaily infraštruktúry naprieč rôznym hardvérom.

CLOUD READY APLIKÁCIE

Sú navrhnuté na prevádzku v cloude. Ich úroveň abstrakcie je nižšia a sú im priznané rôzne detaily infraštruktúry. Cloud týmto aplikáciám dodá príslušné zdroje tak, aby ich aplikácia využila podľa vlastnej potreby. **Celkový výkon aplikácie bežiacej na viacerých serveroch je potom upravovaný pridávaním či uberaním serverov, nie však navyšovaním samotných serverov.** Výpadok časti infraštruktúry rieši samotná aplikácia. Medzi takéto aplikácie patria webové architektúry, pretože sú dizajnovo pripravené na prevádzku v cloude. Následne už záleží len na webovej aplikácii, ako vie tento dizajn využiť.

DOMÁCE ZVIERATÁ A DOBYTOK

Z podobného pohľadu, ako delíme aplikácie na monolitické a cloudové, je možné rozdeliť aj samotné servery. Používa sa na to už dnes známe a zábavné delenie serverov na domáce zvieratá a dobytok.

- **DOMÁCE ZVIERATÁ (PETS)** majú meno a identitu (stav) a každý takýto server vyžaduje osobitnú starostlivosť. V prípade legacy aplikácií je to rovnaké – servery majú svoju identitu. Aplikácia spustená na týchto serveroch nemôže bežať na iných, preto sú do istej miery výnimočné. Ich životný cyklus je typicky dlhý a musia byť zálohované, lebo sú jedinečné.

- **DOBYTOK (CATTLE)** sú hospodárske zvieratá, ktoré nemajú meno ani identitu, sú len číslom v stáde. Neustále vznikajú a zanikajú. Cloud ready aplikácie využívajú tieto servery na rýchle škálovanie, pretože tieto servery nemajú stav. Životný cyklus serverov je často krátky a sú generované podľa potreby. Nie je nutné ich ani jednotlivo zálohovať, pretože ich stačí len nanovo klonovať podľa šablóny. Táto vlastnosť umožňuje agilnú IT infraštruktúru, ktorá sa neustále mení podľa aktuálnych potrieb.

V praxi nie je bežné mať aplikačnú architektúru postavenú výlučne na *pets* alebo *cattle*. Reálna infraštruktúra pod aplikáciou v dnešnej dobe obvykle obsahuje ich kombináciu. Množstvo *pets* serverov zvyšuje cenu a zhoršuje flexibilitu riešenia, pretože *pets* nevedia dostatočne dynamicky reagovať na zmeny. Zvýšená návštevnosť webovej stránky bežiacej na *pets* serveri môže viesť k nútenej potrebe upgradovať server. Ten typicky vyžaduje výpadok a znamená nedostupnosť služieb a administratívnu náročnosť.

CLOUD A VIRTUALIZÁCIA

Cloud nie je len o virtualizácii. I keď fenomén cloudu s virtualizáciou úzko súvisí, nie je jeho jedinou zásadnou a dokonca ani nevyhnutnou technologickou zložkou. Cloud je o sprístupnení rôznych typov IT zdrojov po sieti cez API alebo self-service webové rozhranie. Okrem virtuálnych serverov sú dnes na trhu dostupné aj iné služby vhodné na prevádzku online aplikácií, napríklad:

- BARE METAL DEPLOYMENT
  Cez rovnaké rozhranie, akým sú vytvárané virtuálne servery, je už možné nasadiť aj fyzický server.

- KONTAJNERY
  Sú ľahkou formou virtualizácie, ktorá oddeľuje len aplikácie bežiace pod jedným jadrom operačného systému.

- SIEŤ AKO SLUŽBA
  Možnosť vytvárania rôznych sietí, smerovačov, firewallov a loadbalancerov.

- OBJEKTOVÝ STORAGE
  Služba umožňujúca ukladať súbory ako objekty a sprístupniť ich cez API.

- a mnoho ďalších...

INFRAŠTRUKTÚRA AKO KÓD

Vďaka schopnosti cloudu získavať rôzne IT zdroje cez jednotné rozhranie je dnes možné dodávať IT infraštruktúru softvérovo, to znamená, že jej opis je definovaný v predpise, ktorý hovorí o tom, ako má vyzerať. Servery, siete a iné zdroje sa tak stávajú prvkami programovacieho jazyka.

Nie je teda nutné všetko prácne prepájať, konfigurovať a dokumentovať, ale stačí využiť niektorý z existujúcich nástrojov (alebo ich kombináciu) na automatické vygenerovanie tejto infraštruktúry.

Dôsledkom je aj to, že je možné aplikovať niektoré techniky zo softvérového inžinierstva ako:

- REPRODUKOVATEĽNOSŤ – použitím kódu opakovane získame ten istý výsledok, tú istú infraštruktúru
- ZNOVUPOUŽITEĽNOSŤ – ten istý kód môžeme použiť v rôznych projektoch, a tým skrátiť čas na prípravu novej infraštruktúry
- TESTOVATEĽNOSŤ – kód je možné testovať na rôznych úrovniach od unit testov po integračné testy
- DOKUMENTÁCIA – kód je samoopisný alebo sa z neho minimálne dá vygenerovať referenčná dokumentácia, rieši sa tým problém s aktuálnosťou dokumentácie, ktorá obyčajne existuje vo forme nejakej wiki
- CHANGE MANAGEMENT – možnosť použitia bežných procesov na riadenie zmien počas vývoja softvéru

AGILNÁ IT INFRAŠTRUKTÚRA

Prevádzka aplikácie v internete si vyžaduje mať flexibilnú a agilnú infraštruktúru, ktorou vieme rýchlo reagovať na zmeny.

Internet je dynamické miesto, ktoré sa neustále mení a vplýva na prevádzku aplikácie, ktorá je vystavená do internetu. Nutnosť mať v dnešnej dobe postavenú dôležitú internetovú aplikáciu na agilnej infraštruktúre má viacero dôvodov:

- **NÁRAST PREVÁDZKY** – reakcia na postupné alebo náhle zvyšovanie záťaže by mala čo najmenej zaťažovať samotných administrátorov, aby sa dalo množstvo zdrojov upravovať operatívne a bez väčších znalostí a zásahov operátora.
- **ZMENY V KONFIGURÁCII, VÝVOJ A AKTUALIZÁCIE** – toto v agilnej IT infraštruktúre musí prebiehať tak často, ako sa len dá, aby sa neustále testovali všetky súvisiace procesy. Iba tak je možné dôverovať správnemu fungovaniu systémov aj celého tímu.
- **DDOS/BOTNETY** – automatizované systémy, ktoré skenujú internet v snahe nájsť chyby v aplikáciách alebo uhádnuť heslo do CMS systému. Ich tlak na službu môže byť natoľko silný, že môže dôjsť k úplnému spotrebovaniu zdrojov. Ich útoky si môžu vyžiadať aj rekonfiguráciu systému, aby sa zablokovali nežiaduce požiadavky. Každá aplikácia zverejnená na internete je do niekoľkých hodín vystavená rôznym aplikačným útokom, pretože botnety neustále prehľadávajú celý internet.

AUTOMATIZÁCIA

Prevádzkové požiadavky sa v modernej dobe menia extrémne rýchlo, situácia môže byť iná zo dňa na deň. Preto agilná infraštruktúra nemôže vzniknúť bez automatizovaného konfiguračného manažmentu. Je už nereálne konfigurovať servery ručne a zmeny zapisovať do dokumentov.

Pri ručnej konfigurácii dochádza časom k rozladeniu celého systému a k nečakaným chybám, ktoré spôsobuje ľudský faktor. Len čo dôjde k ručným úpravám systému, narúša sa aj predstava a vedomosti iných členov IT tímu o stave celej infraštruktúry a aj dokonale vedená dokumentácia o konfiguráciách tomu pomáha len čiastočne.

Počet pracovníkov v IT tíme by nemal narastať úmerne s množstvom spravovanej infraštruktúry.

Úlohou konfiguračného manažmentu je tieto zmeny zavádzať do systému a udržiavať ich konzistentné. Ak z akéhokoľvek dôvodu dôjde k odklonu od predpísanej konfigurácie, je nutné, aby systém chybu napravil. Takáto tzv. self-healing architektúra má schopnosť zotaviť sa z veľkého množstva problémov. Príkladom vhodných populárnych nástrojov sú Puppet, CFEngine3 alebo Chef.

Výhodou automatizovanej správy je aj to, že je možné **reprodukovať tú istú infraštruktúru tak často, ako je nutné**. Mnohé virtualizačné riešenia umožňujú beh viacerých prostredí s tými istými IP adresami, takže je možné vytvoriť identické IT infraštruktúry na rôzne účely. Po štarte nového prostredia sa naštartuje aj nástroj konfiguračného manažmentu, ktorý už zvyšok čerstvo vytvoreného systému nakonfiguruje podľa predpisu.

Schopnosť rutinne generovať tú istú infraštruktúru je nesmierne dôležitá aj pri „disaster recovery“ scenároch, keď je nutné, aby na záložnej strane už existovalo prostredie udržiavané rovnako ako produkčné alebo aby ho bolo možné rýchlo vygenerovať. Rozsiahla ručná práca pri „disaster recovery“ situáciách je už dnes skôr neprípustná.

Pomerne novým prvkom automatizácie infraštruktúr je aj ich testovanie. Testovaním vieme odkontrolovať správnosť fungovania konfiguračného manažmentu, či už na existujúcej, alebo na novo vygenerovanej infraštruktúre. V prípade problémov je teda možné ako jeden z prvých krokov spustiť sériu testov na serveri a pokúsiť sa rýchlejšie nájsť problém. Príklad opisovaného testovacieho nástroja je serverspec.org, ktorý je nezávislý od spôsobu, akým bol server nakonfigurovaný, takže je možné ho použiť aj pri ručne nakonfigurovaných serveroch.

TESTOVACIE PROSTREDIA

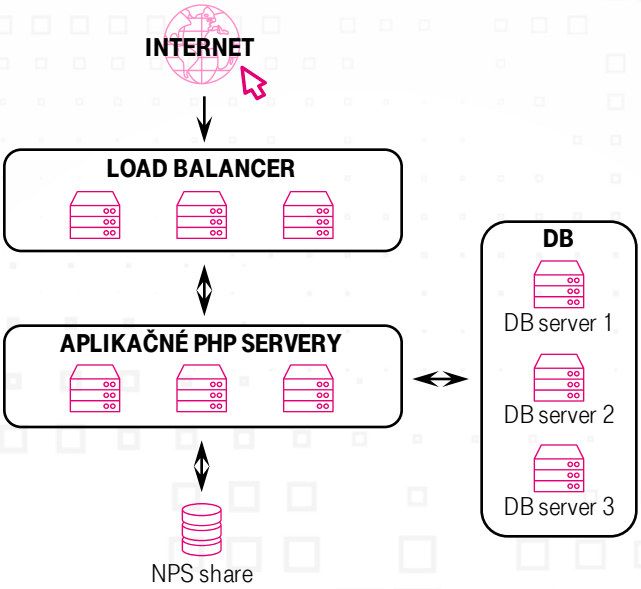
Rôzne testovacie prostredia môžu byť použité na rôzne rozpracované typy zmien. Prvotné produkčné prostredie však musí vždy vychádzať z prostredia testovacieho, aby sa zachoval následný sled zmien. Táto vlastnosť umožňuje vytvorenie aj kvalitného manažmentu zmien, ktorý môže eliminovať väčšinu neželaných dôsledkov zmien.

Testovacie prostredia zvyčajne, žiaľ, len veľmi vzdialene pripomínajú skutočnú prevádzku v reálnom svete. Dá sa argumentovať, že bez automatizovaného konfiguračného manažmentu nie je možné dosiahnuť to, aby testovacie prostredie hodnoverne reprezentovalo to produkčné.

Každá zmena alebo aj vývoj by mali byť podrobené testovaniu v testovacej infraštruktúre a až následne je možné aplikovať ich na produkčné prostredie.

V rámci CI (Continous Integration) procesov je možné generovať testovaciu infraštruktúru napr. každú noc a po jej vygenerovaní spustiť sériu serverspec testov, ktoré potvrdia jej správnu funkčnosť.

TYPICKÁ WEBOVÁ ARCHITEKTÚRA



SKUPINY SERVEROV

V architektúre náročnej webovej aplikácie sa často používa otvorený softvér, ktorý je obvykle na väčšej škále efektívnejší ako komerčne riešenia. V priereze webovej infraštruktúry rozlišujeme nasledujúce vrstvy:

1. Prezentačná vrstva (bezstavová)
2. Aplikačná vsrtva (bezstavová)
3. Backend vrstva (stavová)

PREZENTAČNÁ VRSTVA (BEZSTAVOVÁ)

Prezenčná vrstva je skupina serverov, ktorá je najbližšie k používateľovi a ktorá prezentuje dáta z aplikačného servera. Typicky obsahuje nasledovné funkcie:

- komunikácia cez protokoly HTTP, SPDY, HTTP/2,
- ukončovanie HTTPS spojenia, vo vnútornej sieti sa používa už len HTTP,
- kompresia,
- aplikačný firewall na odfiltrovanie nevhodných požiadaviek smerom na aplikáciu,
- poskytovanie statického obsahu,
- cachovanie statického obsahu,
- smerovanie požiadaviek na konkrétnu skupinu aplikačných serverov,
- rozkladanie záťaže,
- v prípade nedostupnosti aplikačného servera nasmerovanie požiadavky na iný funkčný server,
- v prípade zlyhania spracovania požiadavky preposlanie na iný aplikačný server.

Od týchto load balancerov požadujeme, aby boli schopné zvládnuť čo najviac HTTP spojení, čím dosiahneme plynulejšiu odozvu pre používateľa a zároveň aj odolnosť pred niektorými typmi útokov.

V tejto vrstve je dobré poskytovať statický obsah, ako sú obrázky alebo iné prvky webovej stránky, ktoré sa negenerujú dynamicky. Každú požiadavku, ktorú odchytíme v tejto vrstve, už nie je nutné spracovať na backende, kde to trvá dlhší čas. V tejto vrstve sa aj často cachuje obsah z backendu, aby sa nemusel spracovávať znova. Cachovanie sa robí do RAM alebo na SSD disky, takže odozvy voči klientovi sú nesmierne rýchle a nemenia sa s počtom spojení na load balancer. V prípade úplne nefunkčného backendu môže táto vrstva poskytovať nejaký čas nacachovanú verziu obsahu, a tým znížiť dopady výpadku služby.

Typickými predstaviteľmi týchto load balancerov sú softvéry ako nginx alebo varnish. Nginx sám o sebe možno použiť aj ako klasický webový server, ale využíva sa podstatne častejšie vo funkcii load balancera.

Varnish je známy ako load balancer, resp. HTTP akcelerátor, na čo sa vyslovene špecializuje. Neumožňuje obsah generovať, ale má široké možnosti cachovania, napríklad pomocou technológie ETag, ktorá sa dá použiť aj na cachovanie dynamického obsahu.

Pri každom z týchto load balancerov je možné rozšíriť jeho funkcionalitu cez rozširujúce moduly alebo integrované jazyky, ako je napr. Lua, takže je možné robiť aj komplexnejšie operácie.

Úlohou load balancera je zaistiť aj istú mieru bezpečnosti. Môže odchytávať niektoré nebezpečné požiadavky snažiace sa kompromitovať systém. Prípadne je možné na tejto úrovni zabezpečiť aj GeoIP ochranu a zakázať poskytovanie obsahu vybraným krajinám alebo poskytovať iný obsah v závislosti od krajiny.

APLIKAČNÁ VRSTVA (BEZSTAVOVÁ)

Vrstva aplikačných serverov má za úlohu spracovať požiadavku prichádzajúcu z load balancerov v jazyku webovej aplikácie. Najrozšírenejšie sú jazyky PHP a .NET, množstvo webových aplikácií je však aj v jazykoch Python či Ruby.

Súbory aplikácie bývajú dostupné cez sieťový distribuovaný súborový systém, takže z každého servera sú viditeľné tie isté dáta. Pokiaľ aplikačných dát nie je veľa, je možné ich mať uložené priamo aj na každom aplikačnom serveri, čo však môže rýchlo naraziť na limity, ak množstvo dát, s ktorými aplikácia pracuje, rastie.

Aplikačný server poskytne obsah v súčinnosti s backendom späť klientovi cez load balancer, pričom sa stará aj o cachovanie obsahu tak, aby nebolo nutné obsah nanovo generovať, a tým šetrí zdroje, resp. zvyšuje počet spracovateľných požiadaviek.

BACKEND VRSTVA (STAVOVÁ)

Backend je vrstva, v ktorej už musíme držať stav, napríklad používateľa, alebo konkrétne súbory. Backendom sa môže rozumieť:

- Databázový server – klasický relačný server (MySQL, Oracle, PostgreSQL, iný...), ktorý obsahuje dáta používateľov.
- Storage server – kde sú uložené dáta, ktoré sú cez niektorý sieťový protokol sprístupnené na load balanceroch a aplikačných serveroch.

- NoSQL databázy – veľmi rýchle databázy, ktoré umožňujú ukladať dáta vo formáte kľúč = hodnota priamo v RAM. Typickým predstaviteľom je Redis alebo Memcached. Zvyknú sa sem ukladať medzivýsledky alebo rovno celé výsledky po spracovaní aplikačným serverom, alebo výstupy z databázového servera, čim sa zabezpečí rýchla reakcia.

Spoločným znakom databázových a storage serverov je ich pomerne vysoká náročnosť pri vysokodostupných riešeniach, pri ich škálovaní a správe, pretože tieto vlastnosti nemusia byť implementované v protokole, ktorým sa dorozumievajú.

To sa snažia vyriešiť nové riešenia v podobe nových moderných distribuovaných súborových systémov alebo nového typu databáz ako NoSQL. Pri týchto riešeniach sa ráta s tým, že môžu byť prevádzkované na škále a s výpadkami serverov, preto tieto situácie riešia na úrovni knižníc, ktoré využíva webová aplikácia.

ZÁVER

Doba cloudu so sebou prináša nové prevratné možnosti v spôsobe, akým sú veľké a dôležité weby stavané, vyvíjané a spravované. Je nepopierateľné, že architektúra monolitických aplikácií postavených na jednom serveri nevyhovuje súčasným požiadavkám silnej, dynamickej a bezpečnej online prítomnosti.

Telekom so svojou dcérskou firmou PosAm sú pripravení pomôcť a poradiť zákazníkom postaviť a prevádzkovať vysokoškálovateľnú a dostupnú platformu na prevádzku online aplikácií vhodných pre 21. storočie.

www.TELEKOM**CLOUD**.sk

ZAŽIME TO SPOLU