ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIZERTAČNÁ PRÁCA

Ing. Marek Moravčík

Migrácia služieb pre Cloud Computing

Vedúci práce: doc. Ing. Pavel Segeč, PhD.

Registračné číslo: 28360020183002

Žilina, 2018

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

DIZERTAČNÁ PRÁCA

ŠTUDIJNÝ ODBOR: APLIKOVANÁ INFORMATIKA

Ing. Marek Moravčík

Migrácia služieb pre Cloud Computing

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Školiace pracovisko: Katedra informačných sietí

Žilina, 2018

Poďakovanie

Chcel by som poďakovať svojmu školiteľovi doc. Ing. Pavlovi Segečovi, PhD. za odbornú pomoc, pripomienky a usmerňovanie pri tvorbe práce. Zároveň ďakujem všetkým, ktorí mi pomáhali a podporovali ma.

ABSTRAKT

MORAVČÍK, Marek: *Migrácia služieb pre Cloud Computing*. [Dizertačná práca]. – Žilinská univerzita; Fakulta riadenia a informatiky; Katedra informačných sietí. – Školiteľ: doc. Ing. Pavel Segeč, PhD. – Žilina: FRI UNIZA, 2018.

Dizertačná práca sa venuje problematike Cloud Computingu (CC) z pohľadu portability IaaS (Infrastructure as a Service) riešení. Práca analyzuje súčasný stav v CC, najmä z pohľadu štandardizácie v interoperabilite a portabilite CC riešení. Na základe analýzy súčasného stavu práca navrhuje využiť architektúru MDA (Model Driven Architecture) ako referenčnú architektúru, ktorá je použitá pre návrh transformačných pravidiel pre popis IaaS prostredí a následné riešenie problému portability IaaS služieb. Práca tak ďalej rozpracováva všeobecný MDA model zaoberajúci sa popismi IaaS prostredí a následne zavádza transformačné pravidlá na prepis implementačne závislých prostredí IaaS služieb konkrétnych implementácií (OpenStack vs. Amazon Web Services vs. Microsoft Azure) do všeobecného popisu. Návrh a použitie všeobecného modelu a taktiež proces transformácií ako riešenie otázok IaaS portability je prakticky overený na reálnych implementáciách troch najväčších poskytovateľov IaaS služieb – Amazon AWS, Microsoft Azure a OpenStack.

Kľúčové slová: Cloud Computing, IaaS, Portabilita, MDA, Transformácie.

ABSTRACT

MORAVČÍK, Marek: *Service migration for Cloud Computing*. [Dissertation thesis]. – University of Žilina; Faculty of management science and informatics; Department of information networks. – Supervisor: doc. Ing. Pavel Segeč, PhD. – Žilina: FRI UNIZA, 2018.

This dissertation thesis deals with Cloud Computing (CC) in terms of IaaS (Infrastructure as a Service) portability. It analyses current status in CC, mainly in standardization of interoperability and portability of CC solutions. Based on current status research, the thesis suggests to use MDA (Model Driven Architecture) as reference architecture, which is used for proposal of transformation rules for IaaS environment description and therefore uses them as answer for IaaS portability problem. The thesis elaborates generic MDA model dealing with IaaS environment description and then suggests transformation rules for transcription of implementationally dependent IaaS services description on specific implementations (OpenStack vs. Amazon Web Services vs. Microsoft Azure) to generic description. Suggestion and usage of generic model and also transformation process as answer to portability of IaaS service is practically verified on real implementations of three largest IaaS providers – Amazon AWS, Microsoft Azure and OpenStack.

Key words: Cloud Computing, IaaS, Portability, MDA, Transformations.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

API	Application Programming Interface
ASDK	Azure Software Development Kit
AT&T	American Telephone and Telegraph
AWS	Amazon Web Services
BBC	British Broadcasting Corporation
BMW	Bayerische Motoren Werke
BPMN	Business Process Modeling and Notation
CC	Cloud Computing
CD	Compact Disc
CCIF	Cloud Computing Interoperability Forum
CERN	European Organization for Nuclear Research
CIDR	Classless Inter-Domain Routing
CIM	Computer Independent Model
CLI	Command Line Interface
CPIP	Cloud Portability and Interoperability Profiles
CPU	Central Processing Unit
CWM	Common Warehouse Metamodel
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EaaS	Everything as a Service
EC2	Elastic Compute Cloud
ETSI	European Telecommunications Standards Institute
GPU	Graphical Processing Unit
HDD	Hard Disk Drive
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers

IoT	Internet of Things
IP	Internet Protocol
IT	Informačné Technológie
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JVM	Java Virtual Machine
LAN	Local Area Network
MDA	Model Driven Architecture
MDD	Model Driven Development
MOF	Meta Object Facility
MS	Microsoft
NAT	Network Address Translation
NGN	Next Generation Networks
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NVGRE	Network Virtualization Using Generic Routing Encapsulation
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OS	Operačný Systém
PaaS	Platform as a Service
PHP	PHP: Hypertext Preprocessor
PIM	Platform Independent Model
PKI	Public Key Infrastructure
PSM	Platform Specific Model
RAM	Random Access Memory
RDS	Relational Database Service
REST	Representational State Transfer
S3	Simple Storage Service
SaaS	Software as a Service
SIG	Select Industry Group
SIIF	Standard for Intercloud Interoperability and Federation
SLA	Service Level Agreement

SLDC	Software Development Life Cycle
SOHO	Small Office / Home Office
SQL	Structured Query Language
SRS	System Requirement Specification
SSH	Secure Shell
TCP	Transmission Control Protocol
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDP	User Datagram Protocol
UML	Unified Modeling Language
VM	Virtual Machine
VPC	Virtual Private Cloud
VPN	Virtual Private Network
VXLAN	Virtual Extensible LAN
WAN	Wide Area Network
WG	Working Group
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Úvod

V dnešnom svete informačných technológií (IT) sa môžeme vo viacerých ohľadoch stretnúť s problematikou virtualizácie. Či už ide o virtualizáciu operačných systémov (OS), sieťových prvkov, vývojových platforiem či aplikácií. Virtualizácia takéhoto typu je výhodná hlavne pre poskytovateľov služieb, pretože na jednom fyzickom zariadení môžu poskytovať služby niekoľkým používateľom bez toho, aby tieto služby navzájom kolidovali. Virtualizáciou je možné optimálne riešiť využívanie kapacity fyzických zdrojov so škálovaním a flexibilitou v ich prideľovaní. Koncový používateľa nemá s virtualizáciou žiadnu dodatočnú réžiu, pretože virtuálna služba ktorú využíva má identické správanie, ako služba, ktorá by bola poskytovaná priamo na fyzickom zariadení. V mnohých ohľadoch si používateľ ani nevšimne, že využíva službu, ktorá je virtualizovaná.

Vývoj virtualizácie v IT môžeme datovať od 50-tych rokov minulého storočia, odkedy prešla niekoľkými vývojovými fázami. Prvé náznaky virtualizácie sa objavili v roku 1957, kedy spoločnosť IBM navrhla systém časového zdieľania sálových počítačov. Ďalším významným krokom je koncept virtuálneho stoja (*virtual machine*), ktorý predstavila opäť spoločnosť IBM v roku 1972. Virtualizácia sa postupne vyvíjala, až dospela do dnešného stavu, ktorý poskytuje Cloud Computing (CC) [1].

S pojmom Cloud Computing, označovaný aj „cloud“, sa stretli mnohí používatelia IT, mnohí služby CC využili či využívajú bez toho, aby vedeli, že ide o cloud. Pravdepodobne najznámejšími predstaviteľmi CC sú služby ponúkajúce online úložisko. Patria sem služby ako Dropbox, Google Drive, či Microsoft OneDrive. Ďalším známym predstaviteľom sú kancelárske prostredia s balíkmi používateľského online dostupného softvéru – Google Docs, či Office 365 od spoločnosti Microsoft.

CC prostredia sú využívané v čoraz väčšej miere, v neposlednom rade preto, že významným spôsobom šetria prostriedky, či otvárajú nové možnosti v poskytovaní IT služieb. S rozširovaním povedomia o CC sú si používatelia ako aj poskytovatelia vedomí týchto výhod. CC tak na jednej strane šetrí prostriedky poskytovateľom CC prostredí, pretože optimalizovaným využívaním fyzických zariadení dokážu pokryť požiadavky ďaleko väčšieho počtu používateľov ako tomu bolo pri fyzickom dedikovaní zariadení. Na druhej strane šetria prostriedky používateľom, ktorí nie sú nútení kupovať, prevádzkovať a spravovať mnohokrát finančne náročný hardvér alebo softvér. Poskytovatelia zväčša používajú tzv. „*pay-as-you-go*“ model, kde im používateľ zaplatí len za tie výpočtové zdroje,

ktoré aj reálne spotreboval. Takéto riešenie je obzvlášť výhodné pre väčšie organizácie. Tie tak nemusia spravovať vlastné dátové centrum, a môžu si dynamicky zvyšovať a znižovať prenajatý výpočtový výkon podľa potreby, čo môže pri správnom manažmente v konečnom dôsledku ušetriť nemalé finančné prostriedky.

V súčasnosti sú k dispozícii rôzne CC služby u viacerých poskytovateľov, ktorí ponúkajú rôzne CC prostredia. V nich sú implementujú nové služby, ktoré môžu používateľov presvedčiť o zmene poskytovateľa. Migrácia medzi rôznymi CC prostrediami vytvára problém, ktorý je riešený v tejto dizertačnej práci.

1 Súčasný stav riešenej problematiky doma a v zahraničí

1.1 Úvod do riešenej problematiky

Cloud Computing (ďalej CC) je informačno-komunikačný systém, ktorý ponúka rôznorodé služby na požiadanie. Pojem CC nemá jednoznačnú definíciu. Rôzne organizácie vytvárajú vlastné špecifikácie. Uvedieme niekoľko príkladov. Podľa [1] „Cloud Computing ponúka IT zdroje a aplikácie na požiadanie dostupné cez internet, pričom využíva *pay-as-you-go* model.“ Ďalej v [1] autor uvádza: „Nezáleží na tom, čo si vyberiete, takmer každé CC prostredie má nasledovné charakteristiky: je virtuálne, je flexibilné a škálovateľné, je otvorené (alebo uzavreté), môže byť bezpečné, môže byť dostupné, môže byť bezpečné a dostupné.“ Systém CC je špecifikovaný štandardizačnými organizáciami, ktorých odporúčania sú základom pre štúdium a pochopenie systému CC.

1.1.1 Štandardizácia CC

Cloud Computing je relatívne nové odvetvie IT a nestihlo byť ešte úplne zjednotené a štandardizované. V súčasnosti existuje niekoľko štandardizačných skupín, ktoré sa snažia zjednotiť používanie CC prostredia z pohľadu používateľov a poskytovateľov. Pod pojmom zjednotenia rozumieme poskytnutie jednotného prístupu k CC službám pri rôznych poskytovateľoch. Unifikované prostredie a prístup k nemu je výhodný pre obe strany – používateľa aj poskytovateľa. Používateľ nemusí upravovať svoje aplikácie pri prechode medzi rôznymi CC poskytovateľmi, zatiaľ čo poskytovateľ môže jednoduchšie získať používateľov, ktorí sa rozhodli z nejakých dôvodov opustiť doterajšieho poskytovateľa. Ak má používateľ uspôsobené aplikácie pre chod v CC prostredí, nemusí ich tak nijakým spôsobom meniť, len ich jednoducho presunie k inému poskytovateľovi, prípadne využije služby nového poskytovateľa súbežne so službami aktuálneho poskytovateľa.

V oblasti štandardizácie môžeme štandardizačné organizácie rozdeliť do dvoch skupín. Prvou skupinou sú organizácie, ktoré riešia biznis vzťahy medzi jednotlivými účastníkmi.

Ako príklad organizácie, ktorá rieši biznis štandardizáciu CC prostredia môžeme uviesť európsku komisiu, ktorá vytvorila dve pracovné skupiny - Cloud Select Industry Group (C-SIG) a European Commission Expert group on Cloud Computing Contracts.

Skupina C-SIG európskej komisie má dve pracovné podskupiny. Prvá má názov Cloud Select Industry Group on Service Level Agreements (C-SIG SLA) a zaoberá sa štandardizáciou SLA medzi poskytovateľmi a používateľmi. Taktiež vytvára odporúčania pre správnu formuláciu a vytváranie SLA dohôd. Druhá podskupina má názov Cloud Select Industry Group on Code of Conduct. Tá spolupracuje s inými organizáciami a vytvára a navrhuje zásady správania sa pre poskytovateľov CC prostredí, aby ponúkali unifikované služby a udržiavali dáta používateľov v súkromí [2] [3].

Ďalej je to napríklad European Telecommunications Standards Institute (ETSI), ktorá vytvorila pracovnú skupinu s názvom Cloud Standard Coordination group. Úlohou tejto skupiny je zmapovať aktuálny stav štandardov v oblasti CC, špeciálne v oblasti bezpečnosti, dátovej interoperability a portability [4].

Expert group on Cloud Computing Contracts sa zaoberá zmluvami medzi poskytovateľmi a používateľmi CC prostredí. Jej úlohou je robiť prieskum existujúcich zmlúv medzi používateľmi a poskytovateľmi, získavať najlepšie postupy a na ich základe navrhovať opatrenia na ochranu CC používateľov [5].

Druhou skupinou sú organizácie riešiace prevažne technologické aspekty CC. Uvedieme dve najznámejšie štandardizačné organizácie, ITU-T a NIST. Obidve tieto organizácie vydali štandardy zaoberajúce sa rôznymi aspektami CC systémov. Séria odporúčaní Y v ITU-T sa venuje infraštruktúram NGN (*Next Generation Networks*), IoT (*Internet of Things*) a inteligentným mestám. Odporúčania Y.3500 – Y.3999 sa špeciálne venujú oblasti CC. V ITU-T sú CC venované odporúčania Y.3500-Y.3999 Cloud Computing, v NIST SP 500 a SP 800. Podľa týchto odporúčaní sú ďalej spracované CC špecifikácie.

1.1.2 Špecifikácia pojmu CC

ITU-T v odporúčaní Y.3500 [6] definuje CC ako paradigmu, ktorá cez sieť povoľuje prístup k množine zdieľaných fyzických alebo virtuálnych zdrojov. Tieto zdroje sú podľa potreby škálovateľné a používatelia si ich sami dokážu prispôsobovať. Táto paradigma je zložená z kľúčových charakteristík, používateľských rolí, modelov nasadenia a prierezových CC aspektov.

V NIST SP-500-291 [7] je CC definovaný ako „Cloud Computing je model umožňujúci všadeprítomné, praktické, cez sieť prístupné a na požiadanie dostupné

výpočtové zdroje (napríklad siete, servery, úložisko, aplikácie a služby) ktoré môžu byť rýchlo vytvorené s minimálnym úsilím a bez interakcie s poskytovateľom týchto zdrojov. Tento model je zložený z piatich základných charakteristík, troch modelov služieb a štyroch modeloch nasadenia.“

Podľa uvedených definícií ITU-T aj NIST sú ďalej uvedené podkapitoly Kľúčové charakteristiky, Modely služieb, Modely nasadenia a CC roly a aktivity.

1.1.3 Kľúčové charakteristiky CC podľa NIST

- ***Služba na požiadanie*** *(On-demand self-service)*. Používateľ môže využívať výpočtový výkon poskytovaný CC ako potrebuje, bez toho aby bol požadovaný zásah zo strany poskytovateľa CC služby.
- ***Služba prístupná cez sieť*** *(Broad network access)*. Služby sú dostupné cez sieť za použitia štandardizovaných mechanizmov prístupu cez rôznorodých klientov (osobný počítač, mobilný telefón, tablet).
- ***Spájanie zdrojov*** *(Resource pooling)*. Zdroje poskytovateľa sú agregované, aby mohli slúžiť viacerým používateľom, a sú dynamicky prideľované tam, kde sú aktuálne potrebné. S touto vlastnosťou sa spája aj istá dátová nezávislosť, kedy používateľ nevie, kde sa jeho dáta aktuálne fyzicky nachádzajú.
- ***Škálovateľnosť*** *(Rapid elasticity)*. Zdroje môžu byť dynamicky pridávané a odoberané používateľom podľa potreby, v niektorých prípadoch automatizovane. Pre používateľa sa zdroje javia ako nekonečné, a používateľ tak môže kedykoľvek požiadať o ďalšie, alebo uvoľniť existujúce.
- ***Merateľnosť služby*** *(Measured service)*. CC prostredie aj všetky jeho zdroje majú byť merateľné. Poskytovateľ účtuje službu používateľovi na základe meraní o využívaní zdrojov.

Z uvedených definícií možno konštatovať, že CC systémy sú distribuované technologické platformy, ktoré poskytujú rôzne služby na požiadanie. Preto úspešná implementácia CC systému vyžaduje nielen pochopenie technológie, architektonických vrstiev a modelov, ale aj pochopenie ekonomických a biznis faktorov, ktoré sa nachádzajú v rámci CC prostredí. V nasledujúcej časti práca poskytuje prehľad modelov CC, základných typov služieb a entít, ktoré sú vo väčšine súčasných CC prostredí.

1.1.4 Modely nasadenia CC podľa NIST

Podľa toho, kto môže využívať služby jedného CC prostredia, delíme CC systémy podľa dostupnosti nasadenia na privátny, verejný, komunitný a hybridný systém CC. Identické rozdelenie majú vo svojich odporúčaniach organizácie ITU-T (odporúčanie Y.3500 [6]) aj NIST (odporúčanie SP 500-291 [8]).

Privátny cloud

Privátny cloud je podľa ITU-T Y. 3500 model, ktorý je používaný výhradne jedným CC zákazníkom, ktorý zároveň riadi všetky jeho zdroje. Môže byť riadený buď samotným zákazníkom, alebo treťou stranou, ktorú poverí zákazník riadením. Zákazník môže povoliť prístup do svojho CC prostredia aj iným používateľom.

Podľa NIST SP 500-291 je privátny cloud definovaný ako infraštruktúra určená na využívanie výhradne jednou organizáciou. Táto infraštruktúra môže byť riadená organizáciou, alebo treťou stranou.

Takýto model je vhodný pre organizáciu s dynamickými a meniacimi sa požiadavkami na výpočtové zdroje. Taktiež je vhodný pre organizáciu, ktorá chce využívať výhody CC, ale zároveň potrebuje mať svoje dáta pod kontrolou. To znamená vlastné úložisko v rámci organizácie, ktoré je pod neustálou kontrolou a sledovaním, kto a odkiaľ sa pripája k úložisku.

Privátne CC systémy podľa [9] poskytujú tieto vlastnosti a benefity:

- **Bezpečnosť**
  Kým verejné CC prostredia ponúkajú určitú mieru zabezpečenia, v rámci privátneho CC si organizácia vie limitovať prístup nielen k službám, ale aj k fyzickým zdrojom služby. Takúto limitáciu vie organizácia zabezpečiť pomocou firewallu, prenajatých liniek, či interného hostingu.
- **Absolútna kontrola**
  Keďže sa celé prostredie nachádza v „rukách“ jednej organizácie, táto má dokonalý prehľad o aktivitách, ktoré sa v rámci jej CC prostredia dejú. V prípade vzniku novej požiadavky na CC prostredie vie organizácia veľmi rýchlo reagovať na splnenie takejto požiadavky.

- **Šetrenie a efektivita**

  Pri využití virtualizácie a CC je možné pokryť IT potreby jednej organizácie pomocou niekoľkých fyzických zariadení. Tu nehovoríme len o serveroch a úložiskách, ale aj o záložných zdrojoch či aktívnych sieťových prvkoch. Organizácia tak nepotrebuje toľko administrátorov a školených pracovníkov. Najväčšiu mieru efektivity však môžeme vidieť v poskytovaní prostredí a služieb používateľom na požiadanie, nakoľko každé oddelenie organizácie môže mať rozdielne požiadavky na poskytovaný výkon.

- **Cloud bursting**

  Cloud bursting je nasadenie aplikácie v privátnom CC prostredí, ktoré je oproti iným nasadeniam špecifické v tom, že ak aplikácia potrebuje náhle väčší výpočtový výkon je presunutá do verejného CC prostredia. Následne po čase potreby vyššieho výpočtového výkonu je presunutá späť do privátneho prostredia. Benefit pre používateľa je najmä v tom, že za výpočtový výkon platí poskytovateľovi jen vtedy, keď to aplikácie potrebujú [10].

Organizácia plánujúca nasadiť privátny CC má v súčasnosti na výber z viacerých dostupných riešení. Asi najznámejšie otvorené riešenie (*open-source*) je OpenStack [11], avšak dostupné je aj množstvo iných, napríklad Apache CloudStack, OpenNebula, Eucalyptus a Joyent Triton, OwnCloud. Spomedzi proprietárnych privátnych CC riešení dominujú Microsoft Azure a VMware vSpere, XEN.

Komunitný cloud

Organizácia NIST v odporúčaní SP 500-291 definuje komunitný cloud ako infraštruktúru, ktorá je používaná určitou skupinou (komunitou) so spoločnými záujmami. Podobne ako privátny cloud, aj komunitný môže byť spravovaný komunitou, alebo inou, treťou stranou.

Podľa odporúčania ITU-T Y.3500 je komunitný cloud definovaný ako model nasadenia, kde služby v ňom sú zdieľané medzi skupinu používateľov (komunitou) so spoločnými požiadavkami, kde službu spravuje aspoň jeden člen tejto skupiny. Komunitný cloud môže byť spravovaný komunitou, treťou stranou, alebo ich kombináciou. Na rozdiel od verejného cloudu je komunitný uzavretý výhradne pre komunitu, zatiaľ čo verejný je dostupný komukoľvek.

Podobne ako je privátny cloud využívaný organizáciou alebo jednotlivcom, komunitný cloud využíva nejaká skupina alebo komunita k dosiahnutiu spoločného cieľa. Pod komunitou môžeme rozumieť niekoľko spoločností pracujúcich na spoločnom projekte, alebo skupinu vedcov riešiacu rovnaký problém.

Pri komunitnom CC prostredí je dôležité sa musia používatelia medzi sebou vopred dohodnúť, kto bude spravovať celé prostredie, prípadne kto ho môže kedy a ako využívať. Podľa definície organizácie NIST pri komunitných CC prostrediach je dôležité kto môže pristupovať k prostrediu, nie je dôležité kde alebo ako je celý CC systém nasadený. Akopríklad komunitného CC prostredia býva uvádzaný AWS GovCloud. To je špeciálny, verejnosti uzatvorený priestor v CC prostredí spoločnosti Amazon, ktorý môže využívať len vláda a vládne inštitúcie USA [1] [12].

Komunitný CC systém môžeme nazvať špeciálnym typom privátneho CC systému, pretože viac organizácií využíva jeden spoločný CC systém. Z tohto dôvodu neuvádzame špecifiká komunitných CC systémov, pretože sú identické s privátnymi systémami.

Verejný cloud

Podľa NIST SP 500-291 je verejný cloud definovaný ako infraštruktúra, ktorej využívanie je dostupné širokej verejnosti. Môže byť spravovaná súkromnou organizáciou, akademickou ustanovizňou, vládnou inštitúciou, či ich kombináciou. Táto infraštruktúra sa nachádza v prostredí poskytovateľa.

Podľa odporúčania ITU-T Y.3500, verejný cloud je definovaný ako model nasadenia, ktorého služby sú dostupné akémukoľvek zákazníkovi. Môže byť spravovaný súkromnou organizáciou, vládnou organizáciou, akademickou ustanovizňou, či ich kombináciou. Celé prostredie verejného cloudu sa nachádza v prostredí poskytovateľa služby. Obmedzenia prístupu pre používateľov sú veľmi voľné, niekedy vôbec neexistujú.

Najrozšírenejší typ nasadenia CC v súčasnosti je v súčasnosti podľa [13] verejný cloud. Využívateľmi verejného CC môže byť ktokoľvek s prístupom do siete internet. Poskytovateľov verejného CC prostredia pre organizácie je mnoho, z tých najväčších poskytovateľov sú najznámejší Amazon Web Services (AWS), Google Cloud Platform (GCP) a Microsoft Azure. Avšak podobne ako pri privátnom cloude ich existuje viacero

napr. RackSpace, 1&1, DigitalOcean, Verizon a podobne. Na Slovensku je takýmto poskytovateľom napríklad spoločnosť WebSupport.

Verejný cloud je vhodný pre súkromné osoby požadujúce rôzne typy úložiska pre svoje dáta. Je taktiež vhodný, ak požadujú nejaký druh dedikovaného servera, ako napríklad personálny webový server, ktorý bude nepretržite dostupný. Organizácie zvyčajne využívajú verejný cloud pri outsourcingu IT. Nie je tak potrebné po migrácii IT do verejného CC systému vlastniť žiadny hardvér, čím odpadá nutnosť financovať jeho prevádzku, údržbu a školených pracovníkov, ktorí hardvér spravujú. Verejný cloud je vhodný aj z toho dôvodu, že pokiaľ by organizácia potrebovala krátkodobo vyšší výpočtový výkon, napríklad na riešenie projektu, nepotrebuje kupovať ďalší vlastný hardvér, ktorý by bol neskôr nadbytočný.

Obdobne ako pri privátnych CC systémoch, špecifiká verejných CC systémov by sa dali zhrnúť do niekoľkých bodov [14]:

- **Škálovateľnosť**
  Používateľ môže dynamicky meniť veľkosť svojej infraštruktúry. V prípade potreby vie pridať alebo odobrať aktívne prvky, čím môže pružne reagovať na vlastné požiadavky, alebo na požiadavky svojich používateľov.
- **Šetrenie zdrojov**
  V CC prostredí sa používa model „*Pay-as-you-go*“, čo znamená, že používateľ zaplatí len za službu, ktorú skutočne používa. Ak napríklad prestane využívať mailovú službu, okamžite mu zaniknú náklady na jej prevádzkovanie v CC prostredí. Šetrenie sa však najviac prejaví na tom, že nemusí nakupovať, prevádzkovať a udržiavať vlastnú infraštruktúru a personál potrebný k jej prevádzke.
- **Spoľahlivosť**
  Poskytovatelia CC prostredia dbajú na zálohovanie a bezpečnosť dát svojich používateľov. Zvyčajne je takéto prostredie umiestnené v dátovom centre, kde je väčšina uzlov redundantne pripojená k dátovej sieti, ako aj k elektrickej energii, ktorú v prípade výpadku nahradia generátory.

Pri využívaní verejných CC služieb je potrebné brať do úvahy fakt, že používateľ sa spolieha na poskytovateľa CC prostredia a jeho záruku funkčnosti celého systému. Ak by

poskytovateľ z nejakého dôvodu prestal ďalej poskytovať službu verejného CC prostredia, používateľ by musel hľadať náhradné riešenie pre svoju infraštruktúru. Ďalšou vlastnosťou verejných CC riešení je fakt, že používateľ nemá svoje dáta pod kontrolou. Na úložiskách v priestore CC sa môžu nachádzať aj dôležité a citlivé dáta, pri ktorých nedostupnosti alebo úniku by mohol mať používateľ až existenčné problémy.

Hybridný cloud

ITU-T vo svojom odporúčaní Y.3500 definuje hybridný cloud ako model služby, ktorý pozostáva z aspoň dvoch iných modelov (privátny, komunitný, verejný). Nasadenia týchto modelov sú unikátne, no dohromady tvoria jeden logický celok. Hybridný cloud môže byť spravovaný organizáciou, alebo treťou stranou. Hybridné CC systémy reprezentujú situáciu, kde je potrebná interakcia medzi rôznymi nasadeniami CC služieb, no tieto služby majú rôzne typy nasadenia.

Podľa NIST SP 500-291 je hybridným cloudom nazývaná taká infraštruktúra, ktorá pozostáva z viacerých CC infraštruktúr (privátny, komunitný a verejný CC). Tieto infraštruktúry sú nasadené nezávisle, no zároveň sú prepojené technológiou, ktorá umožňuje dátovú a aplikačnú portabilitu.

Hybridný cloud je kombináciou niektorých z vyššie uvedených CC prostredí. Môže ísť o organizáciu využívajúcu privátny CC pre vlastné účely s tým, že nadbytočný výkon, ktorý nie je schopná sama využiť prenajíma iným subjektom.

Taktiež môže ísť o kombináciu využívania rôznych CC prostredí. Ako príklad môžeme uviesť organizáciu, ktorá využíva privátny CC ako svoju primárnu infraštruktúru, a verejný CC napríklad pre zálohu, alebo testovanie vlastných produktov. Podľa definície organizácie NIST, hybridný CC je akákoľvek kombinácia dvoch alebo viacerých CC prostredí (privátny, verejný alebo komunitný) prepojených technológiou, ktorá poskytuje aplikačnú portabilitu [1]. Keďže hybridný CC systém je kombináciou predchádzajúcich modelov nasadenia, nie je možné určiť jeho špecifiká [15].

1.1.5 Modely služieb CC

Používateľ môže využívať buď len jednu konkrétnu aplikáciu, alebo to môže byť množina aplikácií tvoriaca špecifickú platformu, či môže využívať sieťovú infraštruktúru

poskytovateľa s tým, že si na nej bude prevádzkovať vlastné prostredie. Na základe toho, aké služby a čo môže používateľ využívať, môžeme CC principiálne rozdeliť do niekoľkých modelov služieb. Podľa NIST SP 500-291 [8] poznáme tri základné modely – softvér ako služba (SaaS), platforma ako služba (PaaS) a infraštruktúra ako služba (IaaS). Organizácia ITU-T vo svojom odporúčaní Y.3500 [6] definovala až sedem kategórií. K trom rovnakým ako NIST pridala komunikáciu, sieť, výpočtový výkon a dátové úložisko ako službu. V nasledujúcej časti rozoberieme len kategórie podľa NIST, ktoré sú spoločné aj s odporúčaním ITU-T.

Softvér ako služba (Software as a Service)

ITU-T vo svojom odporúčaní Y.3500 definuje SaaS ako kategóriu služby, kde používateľ má od poskytovateľa k dispozícii aplikáciu. NIST má v odporúčaní SP 500-291 obsiahlejšiu definíciu, kde SaaS je aplikácia poskytovaná používateľovi, pričom táto aplikácia je dostupná z rôznych klientov. Používateľ aplikácie nespravuje CC infraštruktúru.

Pojem Software as a Service (SaaS) sa často spája s biznis aplikáciami a outsourcingom. Nasadením SaaS riešenia odpadá pomerne veľká vstupná investícia do softvérového balíka, ktorý by spoločnosť odpisovala niekoľko rokov. Nehovoriac ešte o potrebe fyzickej a midlvér infraštruktúry, nevyhnutnej pre beh samotnej aplikácie. SaaS aplikácie sú zvyčajne účtované vo forme poplatku na používateľa.

SaaS je z pohľadu koncového používateľa najviditeľnejšia časť CC prostredia. Ovládať takúto aplikáciu môže používateľ buď jej klientskou aplikáciou, no čoraz viac populárnejšie sú prístupy cez webové rozhranie. Benefit webového rozhrania pre používateľa je v tom, že nie je nutné inštalovať klientskú aplikáciu u používateľa, ale stačí webový prehliadač, ktorý sa dnes už štandardne nachádza skoro na každom zariadení či už vo firemných prostrediach, v domácnostiach alebo na osobných zariadeniach. Používanie takejto CC aplikácie je tak platformovo nezávislé, dostupné cez každé zariadenie, pripojené do siete internet. Ako príklad môžeme uviesť Office 365 od spoločnosti Microsoft, ktorá ponúka služby vhodné pre kancelárske prostredie. Veľmi populárne sú taktiež služby úložného priestoru, ako napríklad Google Drive od spoločnosti Google, OneDrive od spoločnosti Microsoft či Dropbox.

Ďalším benefitom pre používateľa je fakt, že celú aplikáciu prevádzkuje jej poskytovateľ. To znamená, že používateľ sa nemusí starať o jej aktualizáciu, zálohovanie

dôležitých údajov a podobne. Táto réžia je na pleciach poskytovateľa aplikácie. Podobne, aby vývojári chceli aplikovať aktualizáciu, či opravu softvéru, stačí ju aplikovať v prostredí dátového centra. Rovnako pri testovaní aplikácie ju nie je nutné testovať na viacerých rôznych platformách.

Niektoré aplikácie alebo výpočty môžu byť náročné na hardvérové prostriedky počítača. Keďže aplikácia beží v dátovom centre, nie je nutné, aby používatelia aplikácie mali výkonný lokálny hardvér. Používateľovi stačí relatívne jednoduchý hardvér potrebný len pre pripojenie do dátového centra na SaaS službu, čo prinesie organizáciám značnú úsporu.

Problémami prevádzkovania aplikácie v CC prostredí sú bezpečnosť a dostupnosť. Niektorí používatelia môžu mať problém s umiestnením aplikácie do CC prostredia, ak im obavy, alebo firemná politika nedovolia umiestniť svoje dáta mimo organizácie. Dostupnosť dát a aplikácií môže byť problém, ak používateľ požaduje neustály prístup ku svojím dátam. Tu môže nastať problém nielen v samotnom dátovom centre, v ktorom je SaaS prevádzkovaná, ale aj na prístupovej ceste, nakoľko sa toto centrum môže nachádzať v geograficky vzdialenej oblasti. Akákoľvek chyba, či u poskytovateľa, používateľa, či na prenosovej ceste znemožní používateľovi pristupovať ku svojím dátam a aplikáciám.

Platforma ako služba (Platform as a Service)

Organizácia ITU-T vo svojom odporúčaní Y.3500 definuje PaaS ako kategóriu služby, kde používateľ má od poskytovateľa k dispozícii platformu. NIST má v odporúčaní SP 500-291 uvedené, že PaaS je možnosť spustiť na CC infraštruktúre používateľskú aplikáciu, pričom zákazník má možnosť využiť programovacie jazyky, knižnice, služby a nástroje poskytované poskytovateľom. Zákazník nespravuje infraštruktúru na ktorej je aplikácia nasadená, no zároveň má plný administrátorský prístup ku svojej aplikácii.

Platform as a Service (PaaS) ponúka používateľovi platformu v CC prostredí, na ktorej si môže používateľ prevádzkovať vlastnú sadu aplikácií, prípadne ju môže využívať ako podpornú platformu pre už existujúce riešenia. Podľa odporúčania NIST môže používateľ na tejto CC platforme vyvíjať a prevádzkovať vlastné aplikácie.

V úlohe podpornej platformy sa najčastejšie stretávame s databázami, ktoré sú spustené v priestore CC. Či už ide o relačné SQL (*Structured Query Language*), alebo

noSQL databázy, PaaS služba prináša používateľovi benefit v možnosti jednoduchej replikácie a zálohovania celej databázy, alebo jej časti [16].

Keďže poskytovateľ necháva používateľom voľnosť nad aplikáciami, zvyčajne sa však nejakým spôsobom chráni pred udalosťami, ktoré by mohli ohroziť jeho infraštruktúru. Medzi takéto prípady patrí napríklad udalosť, keď používateľská aplikácia požaduje čoraz väčšie množstvo zdrojov (CPU, RAM, HDD, ...), čím môže znemožniť funkčnosť nielen iných používateľských virtuálnych aplikácií, ale aj funkčnosť fyzického zariadenia, na ktorom sú virtuálne aplikácie spustené. Poskytovatelia často obmedzujú svojich používateľov použitím tzv. kvót. Kvóty sú horné obmedzenia výpočtových zdrojov (CPU, RAM, HDD) ktoré môžu jednotliví používatelia použiť a na ktorých sa zväčša poskytovateľ dohodne so používateľom pri ustanovení zmluvy o použití CC prostredia.

Infraštruktúra ako služba (Infrastructure as a Service)

Organizácia NIST vo svojom odporúčaní SP 500-291 definuje IaaS ako možnosť pre používateľa vytvoriť si základné výpočtové prostriedky (výpočtový výkon, úložisko, sieťové prepojenie, ...). Používateľ nespravuje infraštruktúru na ktorej sú spustené výpočtové prostriedky, no nad týmito prostriedkami má plnú kotnrolu. ITU-T v odporúčaní Y.3500 definuje IaaS ako kategóriu služby, kde používateľ má od poskytovateľa k dispozícii infraštruktúru.

Služba Infrastructure as a Service (IaaS) je určená pre skúsenejších používateľov, pretože si vyžaduje vedomosti z administrácie operačných systémov. Pri tomto type služby si používateľ kompletne spravuje celú IT infraštruktúru sám, počnúc servermi s ich OS, cez databázy až po sieťové prvky či sieťové prepojenia komponentov. Odporúčanie NIST definuje IaaS ako možnosť pre používateľa vytvoriť a prevádzkovať vlastnú infraštruktúru, na ktorej môže prevádzkovať vlastné aplikácie. Používateľ má k dispozícii základné výpočtové prostriedky (výpočtový výkon, úložisko a sieť), nad ktorými má zároveň administrátorskú kontrolu. Odporúčanie Y.3500 od ITU-T je v definícii strohejšie, hovorí len o tom, že používateľ má k dispozícii od poskytovateľa infraštruktúru.

SaaS, PaaS a IaaS sú tri základné rozdelenia modelu CC služby. V súčasnosti však vznikajú nové a nové modely CC služieb, ktoré poskytovatelia ponúkajú používateľom. Môže to byť napríklad preklad doménových mien - DNSaaS (*Domain Name System as a Service*), firewall - FWaaS (*Firewall as a Service*) či rozkladanie záťaže - LBaaS (*Load*

Balancer as a Service). Vo všeobecnosti môžeme všetky tieto služby zhrnúť jednotným označením „*Čokoľvek ako služba*“ – **XaaS**, alebo **EaaS** (*Everything as a Service*).

1.1.6 Roly v CC

ITU-T vo svojom odporúčaní Y.3500 [6] definovala tri roly, v ktorých účastníci vystupujú – poskytovateľ, používateľ a partner. Poskytovateľ poskytuje používateľom službu, ktorú zároveň spravuje. Okrem ponúkania služby taktiež služby nasadzuje, aktualizuje, monitoruje a podobne. Používateľ je v biznis vzťahu s poskytovateľom, od ktorého odoberá službu. Cloud partner je doplnková rola, ktorej aktivity sa menia v závislosti na vzťahu s poskytovateľom, alebo zákazníkom.

Organizácia NIST v odporúčaní SP 500-292 [7] definuje až päť rôznych rolí. Rovnako ako podľa ITU-T je to cloud poskytovateľ a používateľ, ktoré sú aj rovnako definované. NIST však pridáva tri ďalšie roly. Je to cloud audítor, ktorý vykonáva nezávislú kontrolu nad poskytovanými službami. Ďalej to je sprostredkovateľ služby, ktorý vyjednáva podmienky zmluvy medzi poskytovateľom a používateľom. Poslednou entitou je poskytovateľ prenosu, ktorý poskytuje konektivitu a transport služby od poskytovateľa k používateľovi. Hlbšia analýza sa nachádza v časti 1.2.1.

V nasledujúcej časti uvedieme kompetencie dvoch základných rolí – poskytovateľa a používateľa v troch základných modeloch nasadenia podľa [17] a [18]

Roly v SaaS

Používateľ služby SaaS má dostupné len rozhranie aplikácie, ktorú si u poskytovateľa objednal. Z toho vyplýva, že používateľ nemá možnosť spravovať platformu ani operačný systém, na ktorom aplikácia beží. Používateľ má prístup len k aplikácii, aj to len s používateľskými právami. Môžeme tomu rozumieť tak, že aplikáciu dokáže len používať – nahrať do nej údaje, údaje modifikovať a uložiť. Nemá prístup k aktualizáciám, či modifikácii aplikácie. **Poskytovateľ** CC prostredia má úplnú správu nad hardvérom, operačným systémom a midlvérom, nad ktorým prevádzkuje aplikáciu. Nad aplikáciou má poskytovateľ administrátorskú kontrolu, napríklad dokáže aplikáciu aktualizovať alebo do nej pridávať rozširujúce moduly.

Roly v PaaS

Na rozdiel od SaaS **používateľ** získal prístup k midlvéru, na ktorom prevádzkuje vlastnú aplikáciu, a ku ktorej má úplný prístup. **Poskytovateľ** má aj naďalej úplný prístup k hardvéru a operačným systémom. K midlvéru má administrátorský prístup, teda môže ho aktualizovať, rozširovať a starať sa o správnu funkčnosť. Nad aplikáciou nemá administrátor žiadnu správu, pretože je pod úplnou správou a vlastníctvom používateľa.

Z pohľadu používateľa môže byť platforma nosná časť jeho aplikácie. Ako príklad môžeme uviesť kompilátory a interpretery rôznych programovacích jazykov. Či už je to Java (*JVM – Java Virtual Machine*), .NET alebo Python, používateľ dostane používateľský prístup ku interpreteru jazyka, v ktorom môže spúšťať vlastný kód. Je na zodpovednosti poskytovateľa, aby interpreter či kompilátor jazyka bol aktuálny a pripravený. Za skompilovaný kód a aplikácie bežiace v poskytovanej platforme nesie plnú zodpovednosť používateľ.

Roly v IaaS

Poskytovateľ prostredia nemá žiadnu kontrolu nad zariadeniami používateľa. Pod pojmom kontrola máme na mysli administrátorskú správu. Poskytovateľ môže vytvárať, spúšťať, vypínať, škálovať či mazať zariadenia používateľa. Nemá však do nich žiadny prístup. Poskytovateľ je zodpovedný za funkčnosť hardvéru a hypervízorov, na ktorých sa vykonáva virtualizácia. Poskytovateľ je zvyčajne zodpovedný aj za vysokú dostupnosť IaaS služby a dobrý prístup do svojho dátového centra, v ktorom je služba poskytovaná. **Používateľ** má v správe celú administráciu. Používateľ je tak zodpovedný za plynulý chod aplikácií a celkovú bezpečnosť svojich systémov.

1.2 Referenčné architektúry CC

Referenčné architektúry CC je možné rozdeliť do dvoch skupín. Prvá skupina definuje architektúry CC pomocou rolí (*role-based*). Roly predstavujú entity, ktoré majú vzájomné väzby. Tu patria architektúry:

- DMTF *Cloud Service Reference Architecture*
- IBM *Cloud Computing Reference Architecture*
- NIST *Cloud Computnig Reference Architecture*

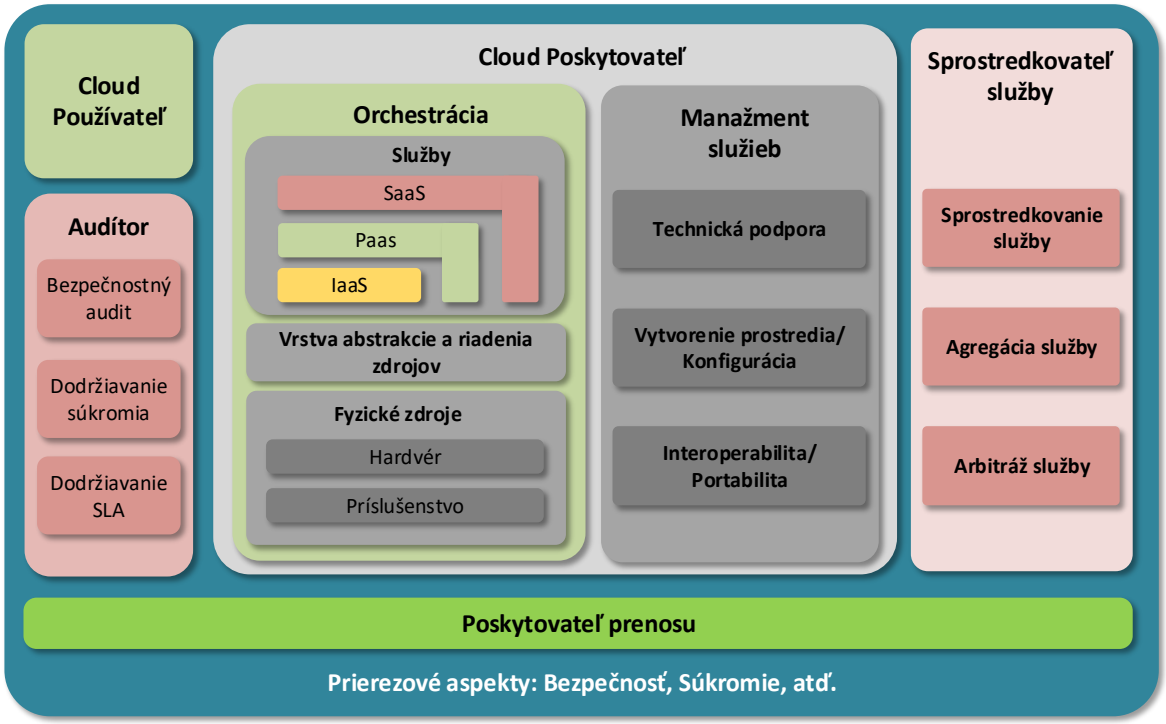
Druhú skupinu tvoria architektúry definované prostredníctvom vrstiev (*layer-based*). V týchto architektúrach sú činnosti mapované do vrstiev, ktoré navzájom spolupracujú. Medzi tieto architektúry môžeme zaradiť:

- CISCO *Cloud Reference Architecture Framework*
- IEFT *Cloud Reference Framework*
- ITU-T *Reference Architecture*

V nasledujúcej časti popíšeme dve architektúry CC systémov, zástupcu role-based (NIST), ako aj zástupcu layer-based (ITU-T).

1.2.1 Referenčná architektúra NIST

Na obrázku Obrázok 1 je znázornená referenčná architektúra CC podľa NIST [7], ktorá určuje hlavné role, ich aktivity a funkcie v CC. Obrázok vyjadruje všeobecnú architektúru na vysokej úrovni abstrakcie. Význam referenčnej architektúry je v tom, že uľahčuje pochopenie požiadaviek, použitia, charakteristík a štandardov CC systémov.



Obrázok 1 Referenčná architektúra podľa NIST, zdroj [7]

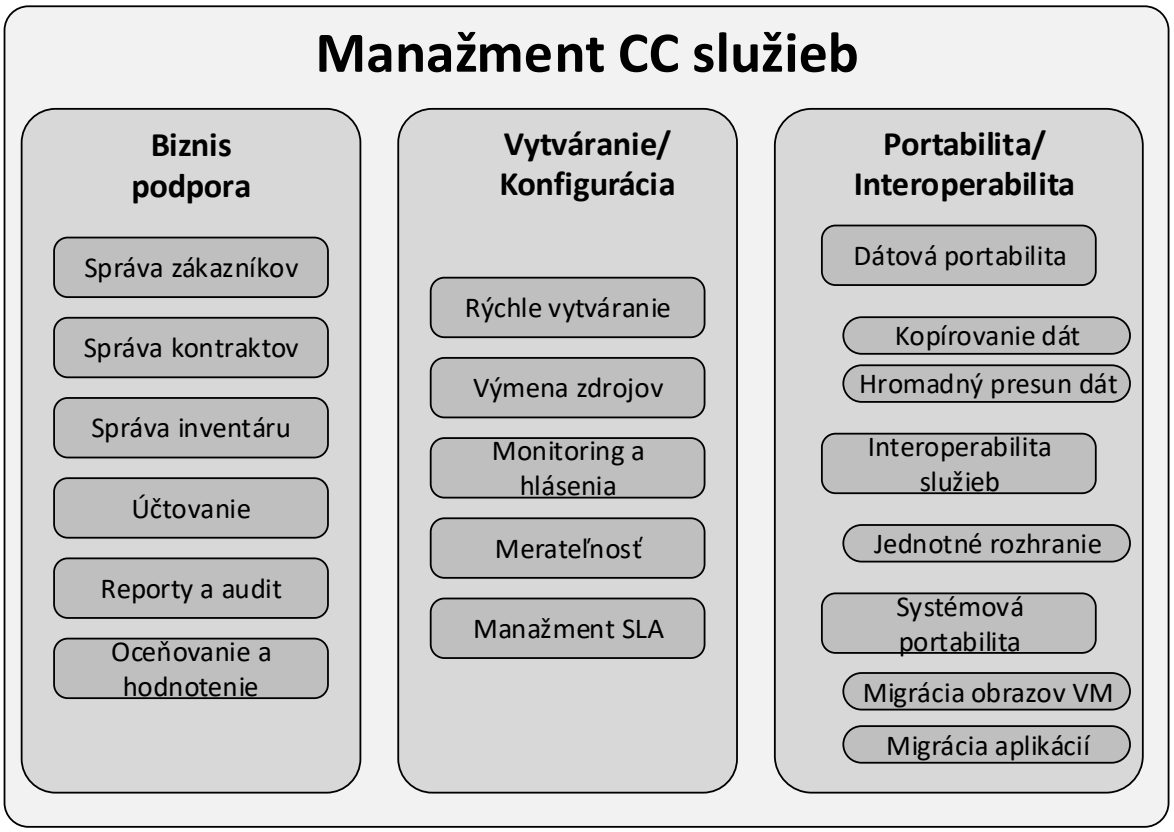
Roly znázorňujú logické postavenie rôznych zainteresovaných strán (*stakeholders*) v rámci CC prostredia, popisujú ich úlohy a povinnosti voči ostatným aktérom. Stručné definície rolí v CC podľa NIST sú v tabuľke Tabuľka 1.

Tabuľka 1 Definície rolí v CC, zdroj [7]

Rola	Definícia
Poskytovateľ (*Provider*)	Entita, ktorá sa stará o chod celého CC prostredia, ktoré ponúka svojím používateľom. Riadi celú infraštruktúru a zodpovedá za funkčnosť a bezchybnosť celého prostredia.
Používateľ (*Consumer*)	Používateľ je osoba alebo organizácia ktorá využíva služby CC. Dohodne sa s poskytovateľom na určitej forme využívania jeho prostredia, ktorá je zvyčajne potvrdená SLA dokumentom.
Audítor (*Auditor*)	Audítor je nezávislá entita, ktorá môže vykonávať kontrolu CC prostredia.
Sprostredkovateľ (*Broker*)	Sprostredkovateľ je entita, ktorá riadi používanie, doručovanie služieb a vzťah medzi konzumentom a poskytovateľom služby.
Poskytovateľ prenosu (*Carrier*)	Entita poskytovateľa prenosu je medzičlánok, ktorý poskytuje prepojenie medzi ostatnými entitami CC prostredia.

Nie všetky roly sú povinné pri využívaní CC služieb, niektoré môžu byť aj zlúčené či vynechané [19] [7] [20] [21] [22].

Pre riešenie témy tejto práce je z referenčnej architektúry NIST, podstatný blok manažmentu CC služieb, znázornený na obrázku Obrázok 2.



Obrázok 2 Manažment CC služieb, zdroj [7]

Prvok „Biznis podpora“ je zameraný na manažment a komunikáciu medzi poskytovateľom a používateľmi. Rozoberá najmä na nefunkcionálne poskytovanie služieb používateľom. Ďalší prvok „Vytváranie a konfigurácia“ špecifikuje vytváranie virtuálnych prostredí a konfiguráciu služby, ako aj vytváranie a manažment výpočtových zdrojov, ich monitoring a prípadné hlásenia porúch či prekročení limitov. Taktiež sa zaoberá manažmentom a dodržiavaním SLA dokumentov.

Poslednú skupinu tvoria „Portabilita a Interoperabilita“. Význam portability a interoperability je hlavne v tom, že ak si chcú poskytovatelia CC udržať aktuálnych a získať nových používateľov, mali by ponúkať mechanizmy podporujúce portabilitu a interoperabilitu CC riešení.

Interoperabilita služieb je schopnosť používateľov využívať služby viacerých CC poskytovateľov cez jednotné rozhranie.

Portabilita je rozdelená do dvoch typov. Pri dátovej portabilite je používateľ schopný kopírovať, alebo presúvať vlastné dáta medzi viacerými CC poskytovateľmi. Systémová portabilita, niekedy označovaná aj ako platformová portabilita, poskytuje možnosť migrácie

inštancie virtuálnych strojov a migráciu aplikácií spolu s ich obsahom medzi poskytovateľmi CC.

Rôzne procesy realizácie CC služieb majú rôzne požiadavky spojené s portabilitou Napríklad pre službu typu IaaS sú to požiadavky na migráciu dát a spustenie služby v novom prostredí. S tým je spojené vytvorenie obrazu aktuálnych VM a ich opätovné spustenie v novom prostredí. Pri službe typu SaaS je pri portabilite potrebné len zálohovať používateľské dáta, zvyčajne v štandardizovanom formáte.

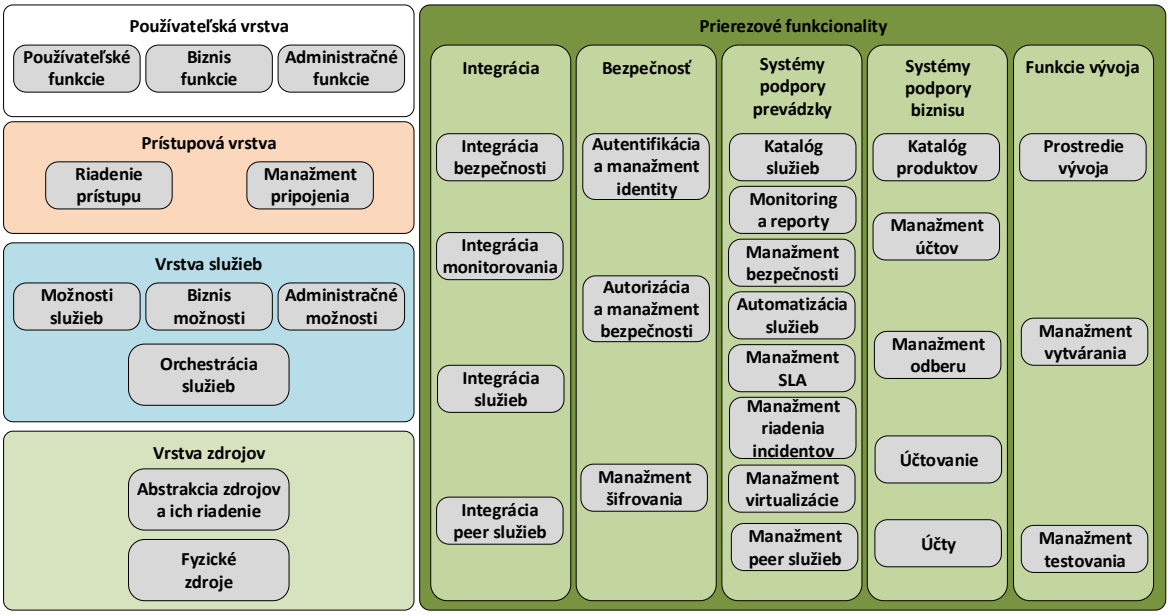
1.2.2 Referenčná architektúra podľa ITU-T Y.3500

Organizácia ITU-T vytvorila referenčnú architektúru CC, ktorú publikovala v odporúčaniach Y.3500 a Y.3502. Táto architektúra je zložená z vrstiev a prierezových aspektov, ktoré nie je možné prideliť jednej vrstve, ale majú priamy dopad na viaceré vrstvy.

Vrstva je definovaná ako množina funkcionalít, ktoré poskytujú podobné funkcie, alebo slúžia podobným cieľom. Odporúčanie definuje štyri základné vrstvy:

- **Používateľská vrstva** obsahuje komponenty podporujúce používanie a interakciu CC prostredia pre CC používateľov.
- **Prístupová vrstva** zabezpečuje svojimi komponentami prístup k jednotlivým funkcionalitám a komponentom CC prostredia.
- **Vrstva služieb** obsahuje komponenty, ktoré ponúkajú služby CC prostredí; komponenty tejto vrstvy taktiež zabezpečujú riadenie a automatizáciu doplnkov potrebných k realizácii služieb.
- **Vrstva zdrojov** obsahuje komponenty, ktoré reprezentujú súčasti a zdroje potrebné k implementácii a prevádzke CC systémov.

Prierezové funkcionality nie je možné jednoducho mapovať do jednotlivých vrstiev, pretože interagujú s viacerými vrstvami. Tieto funkcionality sú bližšie rozobraté v časti 1.2.3. Obrázok 3 znázorňuje celkový logický pohľad na štyri základné vrstvy, aj na prierezové funkcionality referenčnej architektúry ITU-T.



Obrázok 3 Funkcionálne komponenty CC podľa ITU-T Y.3502, zdroj [23]

1.2.3 Prierezové aspekty CC

Odporúčanie ITU-T Y.3502 [23] definuje v CC systémoch prierezové aspekty (*cross cutting aspects*), čo sú vlastnosti týchto systémov, ktoré by mali byť implementované a koordinované na úrovni architektonických pohľadov na CC systém. Tieto aspekty ovplyvňujú CC systémy v toľkých ohľadoch, že ich nebolo možné priradiť ku konkrétnym roliam. Medzi kľúčové aspekty patria:

- **Kontrolovateľnosť (*Auditability*).** Schopnosť zbierať dáta a štatistiky o prevádzke CC systému, na základe ktorých je možné vykonať nezávislé kontroly.
- **Dostupnosť (*Availability*).** Služby CC prostredia by mali byť vždy dostupné autorizovaným entitám. Pod týmito entitami sa zvyčajne myslia používatelia CC prostredia.
- **Interoperabilita (*Interoperability*).** V kontexte CC je to schopnosť používateľa interagovať s CC službou a vymieňať si s ňou informácie. Zvyčajne to znamená, že so službou je možné komunikovať vopred dohodnutým spôsobom. Interoperabilita je taktiež schopnosť jednej služby spolupracovať s inými CC službami. CC používateľ by mal mať štandardizované rozhranie, pomocou ktorého by komunikoval s viacerými CC poskytovateľmi.

- **Údržba a verzionovanie (*Maintenance and versioning*).** Údržba je dôležitá pre správne fungovanie služby. Záleží na type služby, kto má možnosť, resp. povinnosť vykonávať údržbu. Pri službe typu SaaS vykonáva údržbu poskytovateľ, pri IaaS zvyčajne používateľ. Taktiež je podstatné verzionovanie komponentov, prípadne celej služby. Pri CC službách by mal používateľ presne vedieť, akú verziu používa. Či už je to kancelárske prostredie (SaaS), alebo verzia operačného systému (IaaS), rôzne verzie môžu mať funkčný dopad na celkovú stabilitu služby.
- **Výkonnosť (*Performance*).** Aspekt výkonnosti je definovaný ako skupina nefunkcionálnych vlastností CC služby. Patria sem napríklad množstvo výpočtových zdrojov, doba odozvy a celkové oneskorenie pri požiadavkách, či priepustnosť dát cez aktívne sieťové prvky.
- **Portabilita (*Portability*).** Aspekt portability je v CC často požadovaný, pretože používatelia sa zaujímajú o možnosť migrácie svojich dát a aplikácií medzi rôznymi poskytovateľmi s čo možno najnižšími nákladmi a výpadkom služby. Pri službe typu SaaS by si mal byť používateľ schopný presunúť všetky svoje dáta, zatiaľ čo pri službe typu IaaS sú to navyše aj obrazy virtuálnych strojov, ktoré by mali u iného poskytovateľa vykonávať ekvivalentnú službu. V obidvoch prípadoch by mala portabilita poskytovať aj možnosť presunu tzv. metadát, ktoré poskytujú informácie o vzťahu komponentov aplikácie a o potrebnej topológii infraštruktúry (ako napríklad rozkladanie záťaže, firewall a pod.).
- **Regulácia a ochrana osobných údajov (*Protection of personally identifiable information*).** Poskytovatelia by mali dbať na ochranu osobných údajov. V CC je táto úloha o to náročnejšia, že CC systémy často fungujú ako zdieľané prostredia. V mnohých krajinách je takáto ochrana nariadená legislatívne.
- **Odolnosť (*Resiliency*).** Odolnosť je schopnosť systému poskytovať aspoň časť služby aj pri istých výpadkoch systému.
- **Vratnosť (*Reversibility*).** Týmto pojmom by sme mohli označiť proces, v ktorom si používateľ zoberie naspäť všetky svoje dáta, a zároveň poskytovateľ odstráni všetky informácie, ktoré mali spojitosť s používateľom či už na požiadanie používateľa, alebo po uplynutí istého časového intervalu. Tento princíp sa niekedy nazýva „právo byť zabudnutý“.

- **Bezpečnosť (*Security*).** Bezpečnosť pokrýva rôzne oblasti od fyzickej bezpečnosti, až po dátovú bezpečnosť. Radia sa sem techniky ako autentifikácia, autorizácia, nepopierateľnosť pôvodu, dôvernosť, monitorovanie či manažment bezpečnostných politík.
- **Dodržiavanie SLA (*Service level agreement*).** SLA je dohoda medzi poskytovateľom a používateľom CC služby. SLA charakterizuje kvalitu CC služby zvyčajne v technických parametroch, ako napríklad doba výpadku služby a pod.

Mnohé z týchto prierezových aspektov po kombinácii s kľúčovými charakteristikami CC predstavujú dobré dôvody na využívanie CC služieb. Na druhej strane aspekty ako bezpečnosť a ochrana osobných údajov často odrádzajú potenciálnych CC používateľov.

Z uvedeného prehľadu prierezových aspektov sú pre ďalšie riešenie práce podstatné dva prierezové aspekty CC – interoperabilita a portabilita. Pojmy spolu súvisia, ale každý aspekt zabezpečuje rôzne funkcionality CC systémov.

1.3 Interoperabilita CC

Pojem ***interoperabilita*** je vo Wikipédii [24] uvedená takto: „**Interoperability** is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, at present or future, in either implementation or access, without any restrictions.“ Techopédia [25] definuje interoperabilitu ako vlastnosť, ktorá umožňuje neobmedzené zdieľanie zdrojov medzi rôznymi systémami. Môže to byť schopnosť zdieľať dáta medzi rôznymi komponentami cez softvér alebo hardvér. Taktiež to môže byť definované ako výmena informácií a zdrojov medzi počítačmi cez lokálnu (LAN) alebo rozľahlú (WAN) počítačovú sieť. Všeobecne povedané, interoperabilita je schopnosť dvoch alebo viacerých komponentov vymieňať si informácie, a tieto informácie následne vedieť použiť.

V kontexte CC systémov je pojem interoperability uvedený v rôznych odporúčaniach. V [23] je interoperabilita ako jeden z prierezových aspektov špecifikovaná takto: „Interoperability in the context of cloud computing includes the ability of a cloud service customer to interact with a cloud service and exchange information according to a prescribed method and obtain predictable results. Typically, interoperability implies that the cloud service operates according to an agreed specification, one that is possibly

standardized. The cloud service customer should be able to use widely available ICT facilities in-house when interacting with cloud services, avoiding the need to use proprietary or highly specialized software. Interoperability also includes the ability for one cloud service to work with other cloud services, either through a CSP:inter-cloud provider relationship, or where a cloud service customer uses multiple different cloud services in some form of composition to achieve their business goals. Interoperability stretches beyond the cloud services themselves and also includes the interaction of the cloud service customer with the cloud service management facilities of the cloud service provider. Ideally, the cloud service customer should have a consistent and interoperable interface to the cloud service management functionality and be able to interact with two or more cloud service providers without needing to deal with each provider in a specialized way. Standards are implemented in order to support interoperability between components or to support the portability of data or of program components. The implementations should support the evolution of the standards used, both from an earlier version of a standard to a later version, or from one standard to a different one, while minimizing disruptive changes.“

Podľa [26] je všeobecná interoperabilita definovaná ako vlastnosť rozdielnych systémov a organizácií spolupracovať. Interoperabilita v IT a CC je schopnosť výmeny informácií medzi rôznymi systémami a komponentami, a použitia týchto informácií. V [27] je CC interoperabilita uvedená ako schopnosť verejných CC prostredí, privátnych CC prostredí a iných systémov vzájomne zdieľať systémové a aplikačné rozhrania, konfiguráciu, autentifikáciu, autorizáciu, formáty údajov a i. z dôvodu vzájomnej spolupráce.

Interoperabilita je aj témou mnohých článkov, v publikácii [28], z roku 2009, sa autori venujú potrebe štandardizácie CC vo všetkých ohľadoch, aby mohlo byť používané širokou verejnosťou. Zdôrazňujú potrebu štandardov z dôvodu masové využívanie CC, ktorý bude poskytovať stále viac služieb, a vytvorí veľmi komplexné prostredie. Autori upozorňujú, že pri nedostatku štandardizovaných riešení môže nastať prípad, kedy používateľ začne využívať proprietárne služby a nebude môcť v budúcnosti poskytovateľa CC služieb opustiť. Táto situácia je vyššie uvedená pod pojmom „*vendor lock-in*“ a v súčasnosti je motívom pre vytváranie štandardov a jednotných prístupov v rámci CC prostredia.

V článku [29] kolektív autorov poukazuje na dôležitosť jednotného rozhrania ku CC systémom. Navrhujú vytvoriť jednotné rozhranie, UCI (*Unified Cloud Interface*), ktoré

by malo by ponúkať jednotné API pre prístup ku všetkým CC prostrediam. Autori navrhujú rozhranie pre služby PaaS a SaaS, nevenujú sa službe IaaS.

V súčasnosti je viacero organizácií, či pracovných skupín, ktoré pristupujú systematicky k riešeniu interoperability CC. Ako príklad môžeme uviesť organizáciu Institute of Electrical and Electronics Engineers (IEEE), v ktorej v súčasnosti existujú dve pracovné skupiny, CPWG/2301_WG - Cloud Profiles WG (CPWG) Working Group a ICWG/2302_WG - Intercloud WG (ICWG) Working Group.

Pracovná skupina CPWG/2301 [30] pracuje na projekte 2301 – Guide for Cloud Portability and Interoperability Profiles (CPIP), ktorý by mal pomôcť všetkým používateľom CC (používateľom, poskytovateľom a vývojárom) pri vytváraní štandardizovaných rozhraní, formátov a konvencií. Pracovná skupina sa zameriava na komponenty rôznych CC implementácií, ktoré majú často rôzne komunikačné rozhrania a snaží sa navrhnúť také opatrenia, na základe ktorých by mohli vývojári a používatelia unifikovane pristupovať k rôznym implementáciám CC.

Pracovná skupina ICWG/2302 [31] pracuje na projekte 2302 – Standard for Intercloud Interoperability and Federation (SIIF). Tento projekt definuje topológiu, funkcie a kontrolu pri interoperabilite medzi rôznymi CC. K základným elementom projektu patria samotné CC prostredia, brány pre výmenu informácií, menné priestory či rozhrania na audit. Podobne ako CPIP v predchádzajúcom príklade, aj štandard SIIF je stále vo fáze vývoja, a zatiaľ nie je známa jeho finálna podoba.

Ďalším, v súčasnosti uvoľneným štandardom zameraným na CC interoperabilitu je TOSCA (*Topology and Orchestration Specification for Cloud Applications*) [32], ktorú uvoľnila organizácia OASIS (*Organization for the Advancement of Structured Information Standards*) v januári 2014 vo verzii 1.0. TOSCA je otvorené riešenie (*open-source*) jazyk popisujúci vzťahy a závislosti medzi aplikáciami v rámci CC prostredia. Pomocou tohto jazyka vieme popísať rôzne služby, ich komponenty a dokumentovať prostredie v akom sú organizované. To môže byt veľmi užitočné pre administrátorov, ktorí spravujú prostredia v rôznych implementáciách CC, kde pomocou tohto jazyka je ich možné unifikovane popísať a jednotne spravovať.

1.4 Portabilita CC

Pojem ***portabilita*** všeobecne znamená prenositeľnosť produktov alebo komponentov. ITU-T Y.3500 v prierezových aspektoch CC špecifikuje portabilitu ako „Ability of cloud service customers to move their data or their applications between multiple cloud service providers at low cost and with minimal disruption. The amount of cost and disruption that is acceptable may vary based upon the type of cloud service that is being used. ". Ďalej ju rozširuje takto „Portability is significant in cloud computing since prospective cloud service customers are interested in avoiding lock-in when they choose to use cloud services. Cloud service customers need to know that they can move cloud service customer data or their applications between multiple cloud service providers at low cost and with minimal disruption. The amount of cost and disruption that is acceptable can vary based upon the type of cloud service that is being used. For example, if a cloud service customer organization is considering moving from one IaaS cloud service provider to another, the cloud service customer should be able to take its data and the virtual machine (VM) image and get it up and running on an equivalent IaaS service in a relatively straightforward manner. In an SaaS environment, when a cloud service customer organization wants to move an SaaS application to a different cloud service provider (i.e., switch SaaS service providers), the cloud service customer needs to be able to take their data with them, but the rest of the switching cost will include exporting, mapping and importing the data into the new cloud service provider's SaaS application, and that cost is a function of how well the data models and formats of the two SaaS cloud service providers line up. Ideally, SaaS cloud service providers should adopt standard data interchange format(s) relevant to their application domain. Changing between SaaS applications can also involve the cloud service customer adapting to a new service interface (which relates to the interoperability of the service). However, since different cloud capabilities types can have different requirements related to portability, it is more useful to focus on specific types of portability such as cloud data portability and cloud application portability. Cloud service customer data is a class of data objects under the control of the cloud service customer. Cloud data portability allows the cloud service customers the ability to copy cloud service customer data into or out of a cloud service through network access or by physical transfer of storage devices. Cloud application portability allows the migration of items such as a fully-stopped virtual machine instance or a machine image (IaaS service) from one cloud service provider to another cloud service provider, or the migration of application components (PaaS service)

from one cloud service provider to another. In both cases, there is a related aspect of the support of portability of metadata relating to the application components, providing information about the relationships of program components and about the required infrastructure for the program components (e.g., load balancing configuration, firewall settings).“

Ak je interoperabilita definovaná ako komunikácia medzi rôznymi systémami, portabilita je schopnosť používať systémy alebo ich komponenty na rôznom hardvéri alebo softvérových platformách [26]. Portabilita môže byť taktiež možnosť prenosu jednotlivej entity, či celého prostredia bez nutnosti ich modifikácie [33]. O portabilite môžeme uvažovať pri rôznych aplikáciách. Či už je to portabilita softvéru, portabilita elementov v sieti, portabilita telefónneho čísla medzi operátormi, či portabilita nariadení medzi rôznymi inštitúciami.

Používateľ určitej služby tak môže využívať rovnaké funkcie bez ohľadu na to, v akom systéme danú službu využíva. Takýto prístup je pre používateľov veľmi výhodný najmä pri zmene poskytovateľa služby, prípadne pri zmene koncepcie fungovania služby. V CC napríklad pri využití virtualizácie, či priamo CC platformy.

V [7] je portabilita v CC členená na dátovú a aplikačnú. V [34] a [35] portabilitu CC rozčleňujú na dátovú, aplikačnú a platformovú. Popis rôznych typov portability spracovaný ďalej je podľa [27].

Dátová portabilita umožňuje preniesť dáta aplikácií medzi rôznymi implementáciami CC prostredia, prípadne ich preniesť medzi fyzickým serverom a virtuálnym CC prostredím. Takýto prístup je možný, ak sú splnené dve podmienky. Prvá je, že používateľské dáta môžeme exportovať a importovať zo systémov. Export a import sa väčšinou vykonávajú pomocou štandardizovaných rozhraní a protokolov, ktoré väčšina poskytovateľov CC prostredí a operačných systémov ponúka. Druhá podmienka znie, že zdrojový aj cieľový systém musia poznať rovnaký zápis (sémantiku) dát, aby ich boli schopní správne interpretovať. V prípade, že systémy nedokážu spracovať rovnaký formát dát, malo by byť jednoduché dáta medzi formátmi konvertovať bežne dostupnými nástrojmi, bez straty informačnej hodnoty [27].

Aplikačná portabilita umožňuje preniesť aplikáciu, alebo jej časť medzi rôznymi CC implementáciami, prípadne spustiť aplikáciu virtualizovane aj v prípade, že nebola na takýto účel navrhnutá. Za portabilnú aplikáciu sa považuje aj taká, ktorá by mohla vyžadovať

prekompilovanie, prípadne prelinkovanie niektorých knižníc, no v zásade nevyžaduje žiadne väčšie zásahy do svojej štruktúry [27].

Posledným typom je platformová portabilita. V tomto význame je CC platforma podľa [36] definovaná ako infraštruktúra CC služby, ktorá zahŕňa aplikácie umožňujúce používateľom vytvárať a manažovať ich virtuálne prostredia. Platformová portabilita je chápaná ako presun používateľských zdrojov medzi rôznymi CC prostrediami. Zdrojmi sú myslené používateľské dáta nie v zmysle užitočných dát, ako sú napríklad údaje o zamestnancoch či jeho pohľadávkach, ale podporné dáta, ktoré využíva pre fungovanie svojho virtuálneho prostredia. Najčastejším predstaviteľom takýchto dát sú obrazy virtuálnych strojov, ktoré spúšťa vo svojej topológii. Ako príklad môžeme uviesť operačný systém pre webový server. Ak používateľ používa Linux, môže mať pripravený systém s už nainštalovaným webovým serverom, PHP, pomocnými nástrojmi a potrebnými aktualizáciami. Takýto obraz si používateľ môže nahrať do svojho CC prostredia a pri vytváraní nového webového servera je po inštalácii všetko pripravené na spustenie [34] [35].

K platformovej portabilite patrí aj portabilita virtuálnych prostredí. Virtuálnym prostredím sa myslí popis prostredia formou skriptu. Každý významný poskytovateľ CC riešenia ponúka nejakú možnosť, ako skriptom popísať celkové prostredie a entity v ňom. Prostredím rozumieme logickú topológiu siete, IP adresovanie, nastavenie smerovania, DNS a pod. Entity sú spravidla virtuálne stanice (VM), ktoré vykonávajú užitočnú službu pre používateľa (aplikačné servery, firewally, ...). Týmto entitám je možné pomocou vytvorených skriptov nastaviť rôzne údaje, od veľkosti pamäte, jadier CPU, IP adries až po rôzne inicializačné skripty, ktoré sa spustia o prvom nabootovaní VM. Pri rozsiahlejších virtualizovaných prostrediach je práve skriptovanie veľmi často používaným nástrojom administrátorov.

V [27] sú uvedené štyri základné scenáre portability:

1. Používateľ migruje dáta/aplikácie medzi rôznymi poskytovateľmi CC.
2. Používateľ naraz používa viac CC prostredí.
3. Používateľ používa dedikovanú infraštruktúru a CC.
4. Používateľ migruje infraštruktúru do CC.

Pri všetkých štyroch scenároch môže nastať situácia, že používateľ potrebuje presunúť nejakú časť svojej siete. Či medzi rôznymi CC prostrediami, alebo z „*on-premise*“ prostredia do CC, vo veľkej väčšine sa popisy týchto prostredí vytvárajú vo forme nejakého typu skriptu. Rôzny poskytovatelia CC prostredí používajú vlastné zápisy na vytváranie popisu svojich CC prostredí. Používateľ si tak môže pomocou nich vytvoriť topológie a prostredia, aké potrebuje pre svoje aplikácie. Ak by však používateľ chcel zmeniť aktuálneho poskytovateľa CC služby, a požadoval by rovnaké prostredie aj pri novom poskytovateľovi, v súčasnosti nemá inú možnosť, ako vybudovať celé prostredie nanovo pomocou nástrojov, resp. skriptov, v jazyku ktorý ponúka nový poskytovateľ vo svojom CC riešení. Takéto vytváranie je však časovo náročné, najmä pokiaľ ide o väčšie prostredia.

1.5 Špecifikácia problému riešenia

Služby CC sú v súčasnosti na začiatku vývoja a majú mnoho nedostatkov. Jedným z problémov, ktorý označujú mnohí používatelia je „*vendor lock-in*“. Ide o stav, kedy používateľ použije určité rozhranie alebo službu, ktorá je proprietárna len pre daného poskytovateľa. Ak chce používateľ využívať službu iného poskytovateľa, potrebuje signifikantné zmeny svojej služby. Bez zmien musí využívať službu aktuálneho poskytovateľa aj napriek tomu, že iný poskytovateľ ponúka alternatívnu podobnú službu, mnohokrát aj za výhodnejších podmienok.

Riadeniu služieb CC sa venuje skupina štandardizačných organizácií, popísaných v kapitole1.1.4, ktoré sa venujú implementácii služieb CC. Odporúčajú, ako by mala byť riešená spolupráca medzi zainteresovanými stranami, hlavne medzi poskytovateľom a používateľom CC služby. Problém „*vendor lock-in*“ odporúčajú riešiť poskytovaním funkcionalít interoperability a portability.

Interoperabilita, ktorá zabezpečuje vzájomnú komunikáciu platformovo rôznych systémov CC, je základným predpokladom pre využívanie portability. Problematika interoperability je v súčasnosti štandardizovaná a poskytovaná významnými poskytovateľmi CC. Existujúce štandardy sú v súčasnosti dostatočné na poskytovanie elementárnej interoperability medzi rôznymi prostrediami CC služieb, čo umožňuje vytváranie a využívanie CC služieb nad rôznymi prostrediami.

Portabilita znamená migráciu aplikácií, dát alebo systémov medzi platformovo rôznymi CC systémami, respektíve medzi rôznymi CC prostrediami. Mnohé organizácie

sa sústreďujú aj na rôzne aspekty portability uvedené v kapitole 1.4. Ich snahou je prevažne portabilita dát a aplikácií, nie portabilita popisov prostredí. V súčasnosti neexistujú ani odporúčania, ktoré by riešili návrhy konkrétnych technických riešení ako migrovať služby CC medzi rôznymi prostrediami. To znamená presun už existujúceho prostredia alebo inštancií služieb z/do iných prostredí bez nutnosti modifikácie týchto inštancií. Rovnako nie je riešené, ako má byť CC služba implementovaná nad CC prostredím viacerých poskytovateľov.

V kapitole 3.3.1 sú uvedené výsledky prieskumov [37] [38] [13] celosvetových IT spoločností, v ktorých sa počet zamestnancov pohybuje od niekoľkých desiatok, až po tisíce zamestnancov na celom svete. Z prieskumov vyplýva, že vo firemnom prostredí dominuje služba IaaS pred ostatnými CC službami a vo využívaní verejného CC prostredia dominuje spoločnosť Amazon so službou AWS. Nasleduje spoločnosť Microsoft so službou Azure IaaS. Služba IaaS je prevádzkovaná aj na našom pracovisku. Katedra informačných sietí prevádzkuje vlastné IaaS prostredie založené na technológii OpenStack. Služba IaaS je využívaná na vedecké a pedagogické účely. Preto aj riešenie tejto práce bude vzhľadom na komplexnosť všetkých modelov CC služieb zúžené a zamerané na službu IaaS.

Z analýzy súčasného stavu riešenej problematiky môžeme konštatovať, že v súčasnosti neexistuje unifikovaný spôsob transformácie pre migráciu popisu prostredia CC služby IaaS medzi poskytovateľmi rôznych CC prostredí. Nie je preto možné jednoducho riešiť poskytovanie portability IaaS služieb. V dostupnej literatúre nie je popísané žiadne riešenie, ktoré by umožňovalo automatizovane alebo poloautomatizovane transformovať popis implementácie prostredia z jazyka jedného špecifického poskytovateľa do jazyka iného poskytovateľa a tým poskytovať portabilitu CC systému alebo prostredia. Používateľ si môže vytvoriť vlastný nástroj, ktorý by dokázal transformovať popis jeho prostredia, no takýto nástroj by bol len jednoúčelový a viazaný na dvoch konkrétnych poskytovateľov. Preto snahou tejto práce je navrhnúť a vytvoriť všeobecný univerzálne použiteľný mechanizmus, využiteľný pre portovanie IaaS služieb medzi rôznymi CC prostrediami

Tento problém s portabilitou je kritický pre používateľov všetkých CC služieb, nielen IaaS. Existujúce riešenia portability neumožňujú plne využiť výhody rastúceho a meniaceho sa trhu CC a vedie v závislosti od hĺbky využívania služieb k prehlbujúcej sa závislosti na jednom riešení jedného dodávateľa.

2 Ciele práce

Z analýzy súčasného stavu riešenej problematiky a zo špecifikácie problému riešenia uvedeného v kapitole 1.5 môžeme konštatovať, že portabilita v CC službe IaaS je v súčasnosti považovaná za hlavnú prekážku vyriešenia problému *vendor lock-in.* Problém nie je iba štandardizačný alebo legislatívny, ale aj technický. Nie je dostupné univerzálne systémové riešenie popisu prostredia z jazyka jedného poskytovateľa do jazyka iného poskytovateľa. Existujúce jednoúčelové riešenia viazané na dvoch konkrétnych poskytovateľov sú časovo aj cenovo náročné.

Preto sme stanovili hlavný cieľ riešenia práce:

„Návrh systémového prístupu k transformácii CC služby IaaS medzi rôznymi prostrediami a vytvorenie nástroja pre zabezpečenie portability služby IaaS“.

Hlavný cieľ je rozčlenený do štyroch parciálnych cieľov:

1. Návrh transformačných množín v modelom riadenom vývoji - MDD.
2. Zostavenie všeobecného modelu IaaS služby.
3. Vytvorenie transformačných pravidiel medzi všeobecným modelom a najrozšírenejšími CC implementáciami.
4. Praktické overenie funkčnosti navrhnutých transformácií v nástroji pre implementáciu portability virtuálnych prostredí.

Riešenie bude zamerané výhradne na softvérovú portabilitu služby IaaS CC systémov.

3 Metodika a metódy práce

3.1 Metodika riešenia dizertačnej práce

Metodika riešenia vychádza z parciálnych cieľov dizertačnej práce. Etapy riešenia sú zostavené nasledovne:

1. Spracovanie princípov transformácií vhodných k modelom riadenému vývoju použitému pre návrh transformačných množín. Transformácie budú základnými metódami pre návrh transformačných pravidiel.
2. Analyzovanie a vyhodnotenie niektorých prostredí systému CC za účelom získania informácií pre riešenie a overenie nástroja na zabezpečenie portability služby IaaS.
3. Riešenie všeobecného modelu pre popis virtuálnych prostredí. Vytvorenie transformačných pravidiel pre logické entity tvoriace virtuálne prostredie.
4. Pilotná implementácia všeobecného modelu a transformačných pravidiel v zvolenom programovacom jazyku.
5. Overenie implementácie transformačných pravidiel na reálnych CC systémoch.

3.2 Metodológia a metódy riešenia

3.2.1 Model Driven Development - MDD

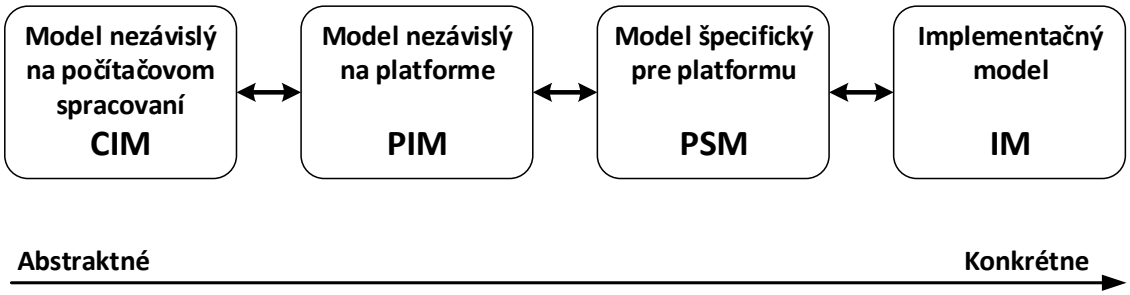
Modelom riadený vývoj (MDD) je založený na paradigme tvorby modelov na rôznych úrovniach vývoja systému. Modely predstavujú abstrakcie, ktorými popisujeme realitu danej úrovne. MDD je často používaný pri vývoji informačných softvérových systémov, čo umožňuje automatizáciu ich vývoja. Dôraz je kladený na aplikáciu princípov vizuálneho modelovania a tvorí osvedčené postupy v životnom cykle vývoja systémov - SLDC. Modely na rôznych úrovniach sú najčastejšie reprezentované kombináciou grafických zobrazení a textu. Textová podoba môže byť v špecifickom modelovacom jazyku alebo prirodzenom jazyku. Medzi príslušnými úrovňami vývoja sú modely transformované, čo umožňuje automatizáciu vývoja. Využitie princípov modelom riadeného vývoja môže byť aj v iných oblastiach.

Najznámejším štandardom pre MDD pre vývoj softvérových systémov je MDA (*Model Driven Architecture*). Významné postavenie má preto, že špecifikuje konkrétne pravidlá pre vývoj softvérových informačných systémov - IS.

3.2.2 Model Driven Architecture - MDA

Modelom riadená architektúra - MDA je jedným z prístupov vývoja softvérového IS, ktorý vyššie spomínané fakty MDD zohľadňuje a reálne uvádza takýto postup do praxe.

MDA je špecifikáciou konzorcia OMG (*Object Management Group*) [39]. Špecifikácia poskytuje návod, ako navrhovať štruktúrované špecifikácie - modely. MDA má špecifikované štyri vrstvy, znázornené na obrázku Obrázok 4.



Obrázok 4 MDA model - rôzne úrovne abstrakcie, zdroj [40]

Počítačovo nezávislá vrstva CIM (*Computer Independent Model*) sa označuje aj ako doménový model, a slúži na popis biznis okolia informačného systému. Z popisu okolia IS sú vytvárané používateľské požiadavky na softvérový systém. Tie sú v súčasnosti spracovávané ako funkčná špecifikácia požiadaviek systému (*System Requirement Specification - SRS*). Je to špecifikácia funkcií, ktoré požaduje biznis okolie IS od softvérového systému. SRS je najčastejšie textová špecifikácia, ale môže obsahovať aj grafické modely biznis procesov, *use case* diagramy, diagramy aktivít alebo stavové diagramy. Modely CIM vrstvy sú nezávislé na technickom riešení. Vytvárajú ich spravidla biznis analytici, prípadne samotní používatelia systému. Ich transformácia do nižšej vrstvy je v súčasnosti predmetom výskumu, ako je napríklad uvedené v [40].

Ďalšia platformovo nezávislá vrstva PIM (*Platform Independent Model*) kompletne špecifikuje funkcionality softvérového systému, ktoré sa transformujú do konkrétnej

implementačnej platformy. Má určitú mieru nezávislosti od konkrétneho riešenia, aby bolo možné použiť akúkoľvek vývojovú platformu. Popisuje algoritmy a štruktúru softvérového systému do takej miery, aby bola prenositeľná a implementovateľná na rôznych technických riešeniach, spravidla v modelovacom jazyku UML (*Unified Modeling Language*). PIM obsahuje informácie, ktoré sú dôležité z hľadiska riešenia a vývoja softvérového systému. Môžu to byť algoritmy, rôzne druhy pravidiel či obmedzení a podobne. Transformácia CIM do PIM v súčasnosti neprebieha automaticky, je riešená podľa úvahy softvérového analytika. Softvérový analytik si z neho vyberie len tie špecifikácie, ktoré sú podľa neho zmysluplné z hľadiska počítačového spracovania daného riešenia. A tu vznikajú problémy, že používateľ má určité funkčné požiadavky, ale tie nie sú riešené ako funkcionality softvérového systému. Táto nepriaznivá situácia je problémom vývoja takmer všetkých IS. Ale každý problém je možné riešiť. Tvrdenie, že to nie je možné lebo je to procesný model a PIM je objektový model neobstojí. Jeden z príkladov je v [40] a je riešený nasledovne. OMG špecifikovala špeciálny grafický jazyk BPMN (*Business Process Modeling and Notation*). Modely BPMN je možné previesť do štandardizovaného platformovo nezávislého formátu XPDL (*XML Process Definition Language*). Platformová nezávislosť a rozšíriteľnosť XPDL formátu reprezentuje štandardizovaný sémantický zápis BPMN notácie, a špecifikácia UML modelov pomocou MOF XMI výmenného formátu sú predpokladom pre CIM – PIM transformáciu. Použitie XPDL a MOF XMI formátu v kombinácii s jazykom XSLT, ktorým sú vytvárané transformačné pravidlá aplikovateľné na popisné formalizmy založené na XML, zabezpečuje platformovú nezávislosť (Linux, Windows) pre automatizovaný prístup k CIM – PIM transformácii. Takouto transformáciou je možné vytvárať modely use case a konceptuálne diagramy tried. Iné riešenia sú uvedené v prehľadovom článku [41]. Platformová nezávislosť PIM umožňuje znovupoužiteľnosť pri rôznych zadaniach, pretože PIM nie je viazaný na žiadnu konkrétnu aplikáciu.

Platformovo špecifická vrstva PSM (*Platform Specific Model*) je už závislá na konkrétnej platforme, na ktorej sa bude systém vyvíjať. Pomocou tohto modelu vieme zo všeobecnej špecifikácie vo vrstve PIM vytvoriť konkrétne štruktúry vo výslednej implementačnej platforme (C++, Java, .Net). Tento model dostatočne odráža štruktúru kódu a je dostatočným podkladom pre implementáciu. Môžeme povedať, že ide o vizualizáciu kódu v konkrétnej vývojovej platforme. Vyskytujú sa v ňom totiž objekty, ktoré sa použijú aj vo výslednom procese programovania, ako napr. triedy, konštruktory, prístup k objektom a pod.

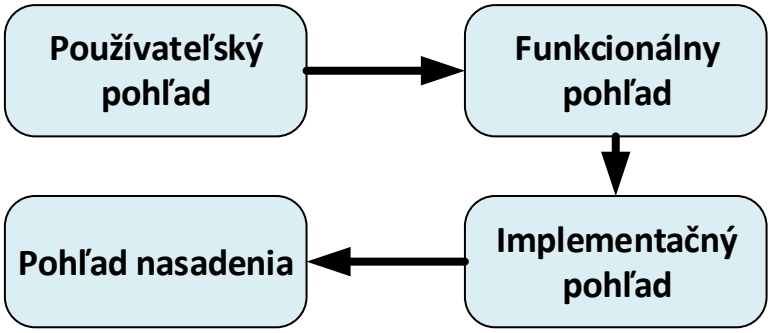
Poslednou vrstvou je implementačný model IM, ktorý predstavuje „kód“. V tejto vrstve sa programuje softvér v zvolenom implementačnom jazyku. Výsledkom tejto vrstvy je skompilovateľný kód, ktorý je pripravený na nasadenie a používanie používateľom. Pokiaľ je potrebné upraviť vybrané funkcionality softvéru, zvyčajne sa spustí celý proces návrhu systému nanovo, teda cez úpravu požiadaviek v CIM vrstve, jej transformáciu do nižších vrstiev, až po zmenu implementačného kódu a jeho opätovné nasadenie u používateľa.

Automatizované transformácie medzi vrstvami PIM - CIM a CIM - PIM sú v súčasnosti funkcionalitami mnohých modelovacích nástrojov.

MDA predpokladá transformácie medzi týmito vrstvami, pričom tvorba vývoja prebieha smerom od najviac všeobecného modelu k modelom viac špecifickým pre platformu, kde sa navrhovaná softvérová aplikácia bude implementovať. MDA princípy môžu byť použité aj reverzne a podľa potrieb iba medzi niektorými vrstvami.

3.2.3 Referenčná architektúra CC

Odporúčanie ITU-T Y.3502 špecifikuje referenčnú architektúru, ktorá poskytuje rámec pre popis CC rolí a sub-rolí a ich aktivít pri vývoji služieb CC. Štyri architektonické pohľady sa zameriavajú na vývoj služieb CC systémov. Ich znázornenie je na obrázku Obrázok 5.

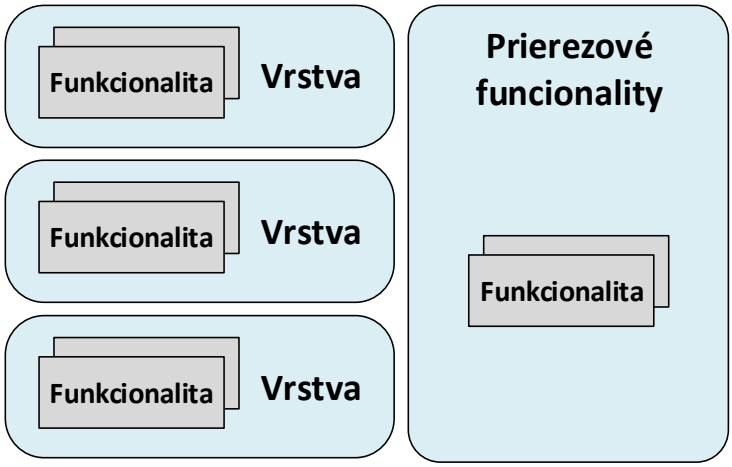


Obrázok 5 Transformácie medzi architektonickými pohľadmi, zdroj [23]

V týchto pohľadoch môžeme vidieť paralelu k štyrom vrstvám princípu MDA. Používateľský pohľad definuje roly, ako sú poskytovateľ, používateľ, a partner. Partnerom

sa rozumie dodatková entita k existujúcim, ktorá podporuje činnosti poskytovateľov alebo používateľov, alebo oboch.

Funkcionálny pohľad je technologicky/platformovo neutrálny, definuje funkcionality potrebné na vytvorenie služby v systéme CC. Koncepcia tohoto pohľadu je znázornená na obrázku Obrázok 6. Tvoria ju vrstvy, komponenty funkcionalít jednotlivých vrstiev a prierezové funkcionality.



Obrázok 6 Funkcionálne vrstvenie CC, zdroj [23]

Funkcionality sú jednotlivé vlastnosti CC prostredia, ktoré sú ponúkané poskytovateľom a používateľom prostredníctvom služieb. Môže to byť riadenie prístupu k zdrojom, automatizácia virtuálnych strojov, monitorovanie výpočtových zdrojov či enkrypcia používateľských dát. Vlastnosti CC prostredí sú ohraničované množinou funkcionalít, ktoré podporujú.

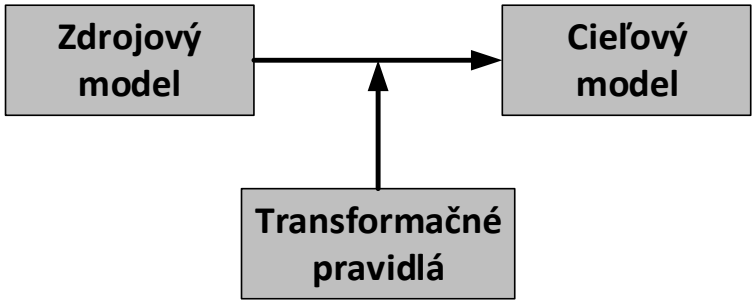
Bližšia špecifikácia implementačného pohľadu a pohľadu nasadenia sa v odporúčaní nenachádza. Je závislá od riešenia konkrétnej služby.

3.2.4 Transformácia a transformačné metódy

Vyriešiť portabilitu systému CC vyžaduje spoluprácu minimálne dvoch zainteresovaných strán služby IaaS. Zvyčajne je to spolupráca poskytovateľa a používateľa. Nástroj, ktorým bude možné transformovať popisy prostredia systému CC bude vzťahom minimálne dvoch rovín, medzi ktorými budú riešené transformačné vzťahy.

Pojem transformácia v MDD má v literatúre viac špecifikácií. Uvedieme citáciu z [42]: „*Transformácia je automatické generovanie výsledného modelu zo zdrojového modelu na základe definície transformácie. Definícia transformácie je množina transformačných pravidiel, ktoré popisujú ako má byť model v zdrojovom jazyku transformovaný do cieľového jazyka. Transformačným pravidlom je popis, ako môžeme jeden, alebo viacero konštruktov v zdrojovom jazyku transformovať do jedného, alebo viacerých konštruktov v cieľovom jazyku*.“

Podľa uvedenej špecifikácie transformácie, zdrojový a cieľový jazyk predstavujú modely. Transformácia je proces, ktorým je zdrojový model konvertovaný do cieľového modelu prostredníctvom transformačných pravidiel. Proces transformácie je na obrázku Obrázok 7.



Obrázok 7 Transformačné pravidlá, zdroj [40]

Na definovanie transformačných pravidiel môžu byť použité rôzne metódy. Dostupná literatúra uvádza viac prístupov alebo princípov, na základe ktorých môžeme vytvoriť transformačné pravidlá pre rôzne modely. Uvedieme niektoré z nich.

Špecifikácia Common Warehouse Metamodel (CWM) [43] konzorcia Object Management Group (OMG) uvádza koncept transformácií ako „čierna aj biela skrinka“. Systém čiernej skrinky transformuje modely pomocou zabudovaných mechanizmov, ktoré nie sú používateľovi známe, a nemôže ich ani upravovať. Pri systéme bielej skrinky si môže používateľ definovať, ako majú byť prepojené zdrojové a cieľové elementy pomocou *ProcedureExpression*.

Iným prístupom je transformácia grafových modelov popísaná v [44] a [45]. Táto transformácia pozostáva z množiny pravidiel, ktoré sa riadia najmä matematickými

princípmi a číselnými operáciami. Je určená predovšetkým pre matematické modely. Princíp je v tom, že zo zdrojového modelu - grafu vytvára podgraf, na ktorý je možné niekoľkonásobne aplikovať pravidlá, až kým nedosiahneme požadovaný výsledný graf.

Koncept ktorý používa transformáciu XML dokumentov sa nazýva XSLT (*Extensible Stylesheet Language Transformations*) [46] a bol špecifikovaný konzorciom W3C. XSLT je jazyk určený na transformáciu XML dokumentov, kde zvyčajne výstupom je taktiež XML dokument, no výstupom môžu byť aj iné dokumenty, napríklad HTML či PDF.

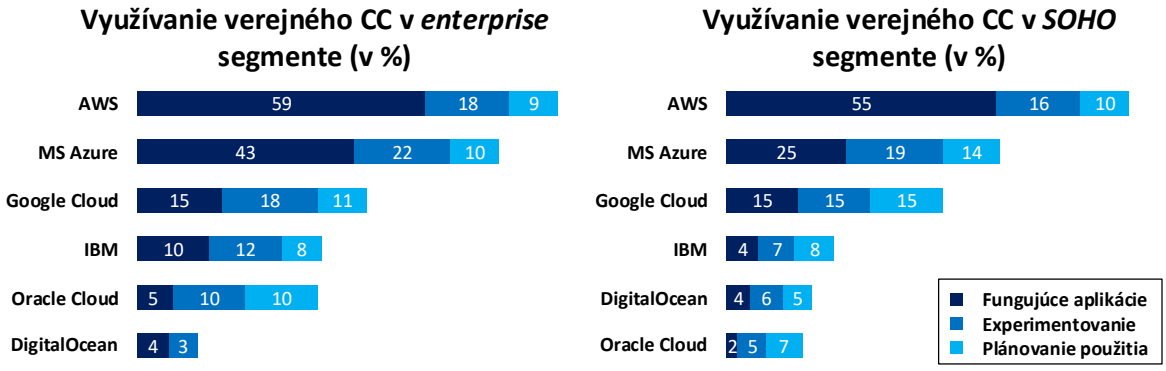
Princípy a metódy uvedené v tejto kapitole tvoria základ ďalšieho riešenia práce.

3.3 Prehľad CC IaaS riešení

3.3.1 Využívanie IaaS

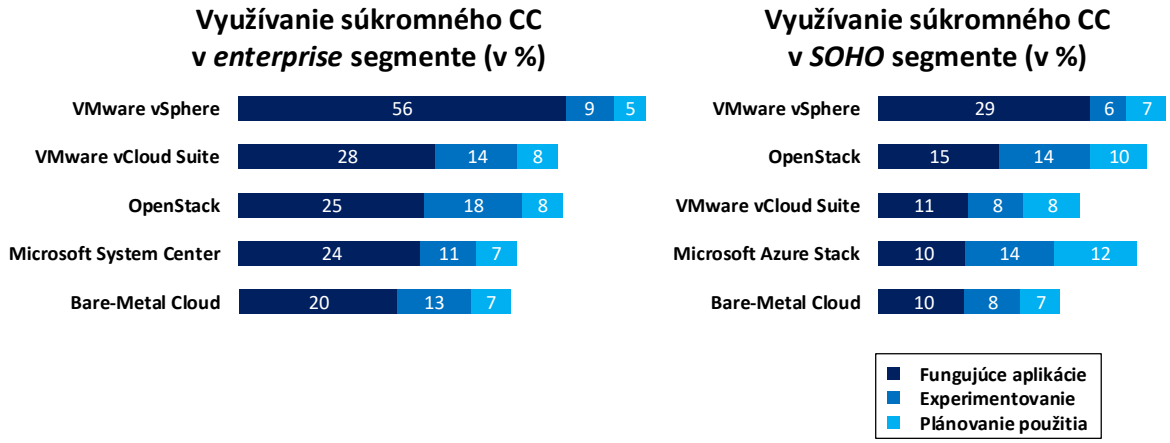
Z poskytovaných CC služieb sa zameriavame v práci hlbšie na službu IaaS. Dôvodom je tá skutočnosť, že spomedzi základných CC služieb (SaaS, PaaS, IaaS) je v podnikovom prostredí najrozšírenejšia práve služba IaaS. Vyplýva to aj z prieskumov [37] [38] a [13], kde medzi organizáciami rôznej veľkosti dlhodobo dominujú práve riešenia ponúkajúce primárne IaaS službu. Vyplýva to aj na obrázku Obrázok 8, ktorý znázorňuje využívanie verejných CC služieb v podnikovom prostredí z roku 2017.

Suverénne najväčší podiel na trhu verejného CC má dlhodobo spoločnosť Amazon s produktom AWS, nasledovaná riešeniami od spoločností Microsoft a Google.



Obrázok 8 Využívanie verejného CC organizáciami, zdroj [13]

Podľa reportu uvedeného v [13] rebríček najpoužívanejších privátnych CC prostredí pre rok 2017 vedie spoločnosť VMware, nasleduje OpenStack, a Microsoft Azure. Percentuálne zastúpenie na trhu je znázornené na obrázku Obrázok 9. Rovnako ako v prípade verejných CC prostredí, aj najpoužívanejšie privátne CC ponúkajú primárne službu IaaS. Preto sa v nasledujúcej časti zameriame na stručný prehľad najpoužívanejších IaaS CC riešení, ktoré sa v súčasnosti používajú vo veľkých podnikových (*enterprise*) ako aj malých SOHO (*Small Office/Home Office*) organizáciách.



Obrázok 9 Využívanie privátneho CC organizáciami, zdroj [13]

3.3.2 Amazon Web Services

Najpoužívanejším verejným CC prostredím je Amazon Web Services (AWS). V roku 2015 a 2016 ho používalo až 57% používateľov, ktorí využívali služby verejného CC poskytovateľa [13]. Systém bol oficiálne spustený v roku 2006 ako služba pre hosting webových služieb, a už v roku 2007 v ňom pracovalo viac ako 180 000 vývojárov. V súčasnosti (marec 2017) sa jeho dátové centrá rozprestierajú po celom svete a sú združené do 16-tich regiónov [47].

Medzi základné časti systému AWS patria moduly EC2 (*Elastic Compute Cloud*) a S3 (*Simple Storage Service*). EC2 je modul zabezpečujúci výpočtový výkon, narábanie s obrazmi virtuálnych systémov a sieťové pripojenie virtuálnych prostredí. Modul S3 zabezpečuje úložný priestor pre virtuálne VM inštancie. S3 však neponúka úložný priestor vo forme databázy, ale len ako blokové úložisko. Relačné databázy ponúka modul RDS (*Relational Database Service*). Samozrejmosťou pri systéme AWS je rozsiahla online dokumentácia podrobne popisujúca všetky moduly, ktoré môže používateľ využiť. AWS

ponúka možnosť využiť skripty na vytváranie a modifikáciu virtuálnych prostredí. V prostredí AWS sa tieto skripty nazývajú CloudFormation.

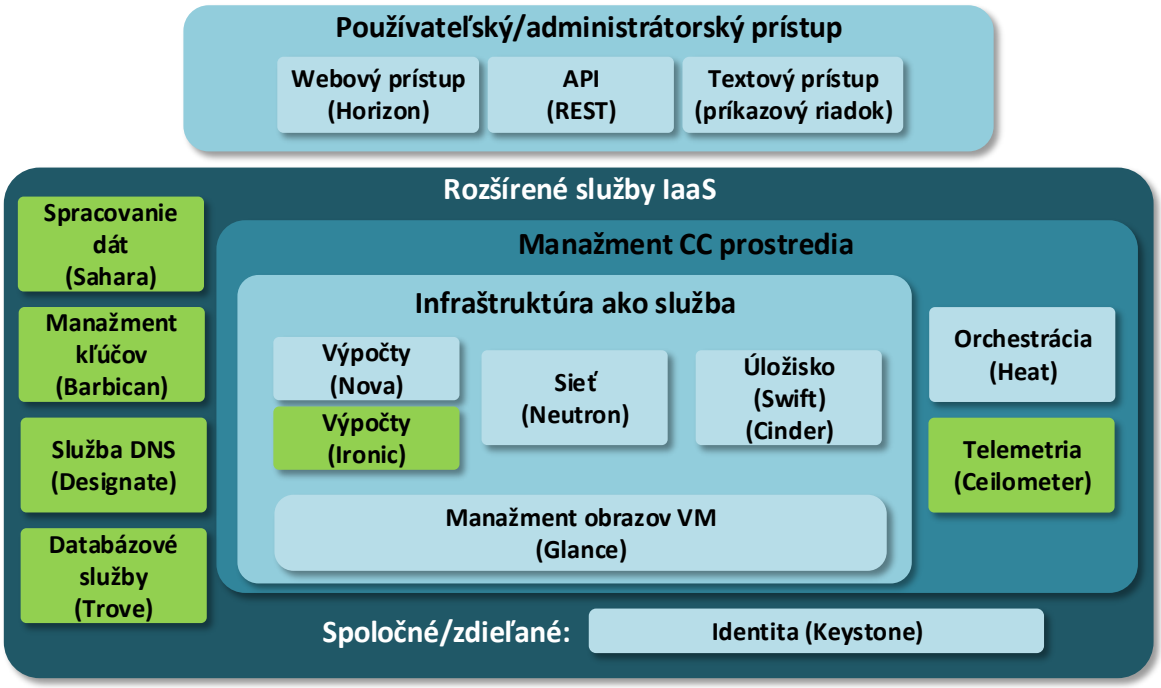
3.3.3 OpenStack

Systém OpenStack je CC platforma patriaca do rodiny otvorených riešení (*open-source*) vydaná pod licenciou Apache License 2.0. Je druhou najčastejšie využívanou privátnou CC platformou pre prevádzku IaaS služieb v Enterprise a Small business prostredí. V súčasnosti (marec 2017) v tomto segmente dominuje riešenie VMware vSphere/vCenter [13]. Projekt OpenStack iniciovali v roku 2010 spoločnosti NASA a Rackspace Holding. Od roku 2012 je pod záštitou neziskovej organizácie OpenStack Foundation [11].

Podľa prieskumov sa OpenStack stal najpoužívanejším open-source CC riešením, a medzi jeho používateľmi sú významné spoločnosti, ako napríklad AT&T, BBC, BMW, CERN, eBay, PayPal či NSA [11]. Jedným z dôvodov, prečo sa OpenStack rozšíril je pomerne veľká komunita vývojárov a používateľov, ktorí neustále vylepšujú systém. Nová verzia je pravidelne uvoľňovaná každého pol roka, na jar a na jeseň. V súčasnosti (marec 2017) je posledná pätnásta verzia, s kódovým označením Ocata. Podobne ako AWS, aj OpenStack má rozsiahlu online dokumentáciu a možnosť využívať skripty automatizácie. Modul zabezpečujúci automatizáciu sa nazýva Heat, a podľa neho majú orchestračné skripty meno HOT – Heat Orchestration Templates. Okrem skriptov je možné využiť aj priame volania funkcií systému cez štandardizované rozhranie REST API, či priamo cez príkazový riadok riadiaceho uzla celého systému.

Systém OpenStack je modulárny. Používa niekoľko povinných modulov nevyhnutných pre základnú prevádzku systému (Keystone, Glance, Nova a Neutron). K nim je možné pridať množstvo doplnkových modulov, podľa potreby a priania používateľov. V súčasnosti si je možné vybrať celkovo z 19 modulov [48]. Moduly, ako aj samotná koncepcia systému, je výrazne inšpirovaná AWS prostredím. Aj keď AWS má niekoľkonásobne viac služieb, ak spustíme systém OpenStack so základnými modulmi, nájdeme v nich oproti AWS minimálne rozdiely. Na obrázku Obrázok 10 sú znázornené najčastejšie používané moduly pri nasadení systému OpenStack, kde názov modulu sa nachádza v zátvorke. Modrou farbou sú označené povinné moduly a tie, ktoré

sa nachádzajú v takmer každom nasadení. Zelenou farbou sú znázornené moduly, s ktorými sa stretávame zriedkavejšie pri nasadeniach systému.



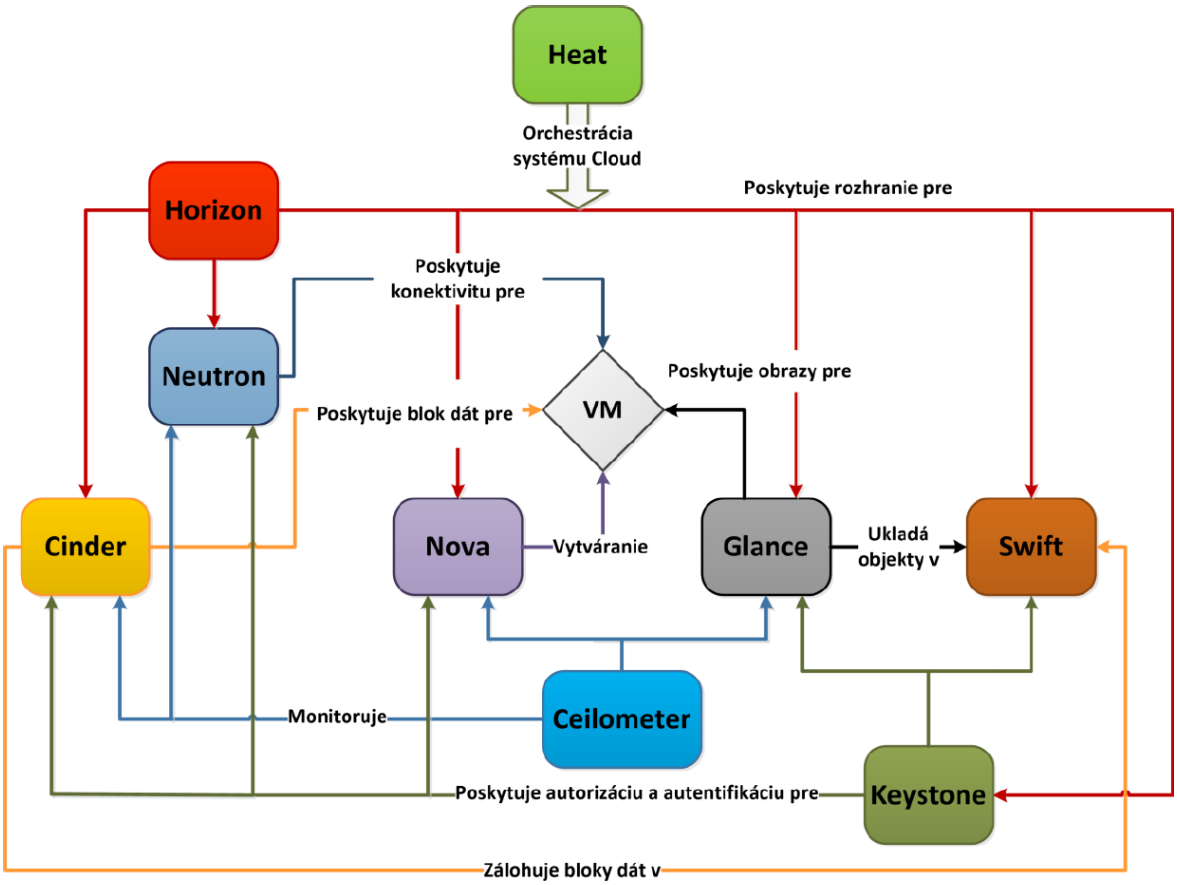
Obrázok 10 Základné moduly systému OpenStack, zdroj [49]

Koncepcia nasadenia systému OpenStack sa skladá z riadiaceho uzla (v terminológii nazývaný *controller*), výpočtových uzlov (nazývaných *compute nodes*) a úložiskových uzlov. Riadiaci uzol zabezpečuje komunikáciu všetkých uzlov navzájom, a riadi činnosť všetkých modulov na všetkých fyzických zariadeniach, ktoré tvoria OpenStack nasadenie. Aj keď je systém OpenStack navrhnutý na distribuované nasadenie, teda na viacerých fyzických zariadeniach (serveroch), nie je to vždy nutné. Ak je plánované len jednoduché nasadenie, prípadne overenie funkčnosti, je možné všetky moduly nainštalovať na jeden server, kde bude systém plnohodnotne pracovať. Pri rozsiahlejších nasadeniach sa však odporúča rozkladať záťaž a použiť viacero fyzických zariadení pre jedno nasadenie. V niektorých prípadoch je dokonca nutné použiť niekoľko fyzických serverov, ak sú virtuálne prostredia príliš veľké pre jeden server.

V nasledujúcej časti stručne rozoberieme štyri základné moduly systému OpenStack (Keystone, Glance, Nova a Neutron), ktoré sú prítomné pri každom nasadení systému. Všetky ostatné moduly sú závislé na službách aspoň jedného z týchto modulov. Dôvodom

popisu je tá skutočnosť, systém OpenStack sme nasadili na Katedre informačných sietí do prevádzky pre experimentálne účely tejto práce. V súčasnosti (marec 2018) ho prevádzkujeme aj pre vedecké a pedagogické účely. Máme s jeho používaním praktické skúsenosti.

Konceptuálny model systému OpenStack sa nachádza na obrázku Obrázok 11, na ktorom je zobrazené prepojenie a závislosti najčastejšie nasadzovaných modulov v systéme OpenStack.



Obrázok 11 OpenStack - konceptuálny model riešenia, zdroj [19]

Prvým modulom, ktorého funkcie využíva celý systém, je Keystone. Tento modul ponúka funkcie autentifikácie a autorizácie. Pri inštalácii akéhokoľvek ďalšieho modulu je tento nový modul nutné konfigurovať tak, aby bol schopný komunikovať s modulom Keystone. Pokiaľ táto komunikácia nie je zabezpečená, systém ako celok odmietne nový modul používať. Okrem autentifikácie modulov sa Keystone stará aj o vytváranie projektov, a používateľských účtov. Pri prihlasovaní používateľa do systému, modul Keystone

autentifikuje a autorizuje používateľa a pridelí mu tzv. token, pomocou ktorého môže využívať funkcie ostatných modulov. Modul Glance je zodpovedný za vytváranie a uchovávanie obrazov virtuálnych strojov (VM – *Virtual Machine*), z ktorých sa neskôr klonujú virtuálne inštancie v rámci CC prostredia.

Ďalším základným modulom je Nova. Nova zabezpečuje vytváranie a prevádzku VM. Pri vytváraní VM riadiaci uzol vydá pokyn pre niektorý z výpočtových uzlov na vytvorenie VM z existujúceho obrazu. Nova sa stará o beh samotnej VM, pridávania a odoberanie zdrojov, ako napr. CPU, RAM, HDD a pod. Úzko spolupracuje s modulom Neutron, ktorý je zodpovedný za časť siete. Neutron vytvára virtuálne siete pre VM, ale taktiež zabezpečuje pokročilé funkcie, ako napríklad Virtuálne privátne siete (VPN), preklad adries NAT či firewalling.

Môžeme povedať, že štyri uvedené moduly dokážu zabezpečiť základnú funkcionalitu IaaS služby v systéme OpenStack. Veľmi často sú však tieto moduly doplnené inými, ktoré zvyšujú funkcionalitu, ale aj komfort používania CC systému OpenStack. Zo všetkých spomeňme aspoň modul Heat. Heat sa stará o orchestráciu, teda automatizáciu používania CC systému, a je postavený na báze skriptov. Je veľmi často nasadzovaný z hľadiska jednoduchšieho používania celého systému. Práve možnosti pokročilej automatizácie sú jeden z faktorov, ktoré sú pri CC riešeniach veľmi zaujímavé.

3.3.4 Microsoft Azure IaaS

Oficiálne spustenie platformy MS Azure bolo 1. februára 2010, vtedy ešte s pôvodným názvom Windows Azure. Na Microsoft Azure bola platforma premenovaná v marci 2014. Podobne ako AWS, aj Azure je rozmiestnený v mnohých dátových centrách, ktoré tvoria 34 regiónov, kde si môžu používatelia prevádzkovať vlastné prostredia [50].

Azure na svojich webových stránkach uvádza, že ponúka stovky služieb, no je to len jemnejšia granularizácia rovnakých služieb, ako ponúkajú OpenStack, či AWS. Je to výpočtový výkon, rôzne typy databáz (relačné aj nerelačné), rôznorodé využitie kontajnerových služieb, sieťové služby ako rôzne typy virtuálnych sietí, VPN siete či DNS [50].

Samozrejmosťou je ponúkanie automatizačného nástroja. Opäť sú to orchestračné skripty, ktoré sa píšu v skriptovacom jazyku vyvinutým spoločnosťou Microsoft, v jazyku PowerShell. Tieto skripty sa nazývajú „Runbooks“. Používateľ má na výber z niekoľkých

desiatok predpripravených skriptov v repozitári „Azure Automation Runbook Gallery“, alebo si môže vytvárať vlastné. Veľkým benefitom hlavne pre začínajúcich administrátorov je fakt, že Azure ponúka aj grafické prostredie, kde si používatelia môžu systémom „drag-and-drop“ poskladať vlastný runbook bez nutnosti ovládania jazyka PowerShell.

V júli 2017 ponúkla spoločnosť Microsoft systém Azure aj na voľné stiahnutie a experimentovanie. Edícia sa volá Azure Stack Development Kit (ASDK), a je to systém určený na preskúšanie a otestovanie platformy Azure, ktorý je nainštalovaný v dátovom centre používateľa. ASDK má obmedzenie v tom, že ho je možné nainštalovať len na jeden fyzický server [50] [51] [52].

3.3.5 VMware vSphere

Podľa [13] je VMware vSphere najpoužívanejšou platformou v privátnych CC prostrediach. Za svoje rozšírenie vďačí najmä masívnej technickej podpore zo strany spoločnosti VMware. Platforma vSphere je komerčný produkt, ktorý je licencovaný podľa počtu CPU, na ktorých bude bežať hypervízor. Platforma vSphere používa vlastný hypervízor – ESX/ESXi. VMware vSphere je možné zakúpiť v rôznych verziách, ako napríklad essential, standard, či enterprise. Tieto verzie sa líšia podľa stupňa výbavy, ktorú môže používateľ využívať. Najmä pri vyšších verziách a viacerých fyzických serveroch je vhodné používať nástroj VMware vCenter, ktorý centralizuje manažment jednotlivých fyzických uzlov [53].

Už v základnej verzii je možné využiť automatizáciu, podobne ako v platformách AWS, či Microsoft Azure. Komponent ponúkajúci automatizáciu sa nazýva VMware vRealize. Nástroj vRealize umožňuje jednoduchým spôsobom vytvoriť popis prostredia a virtuálnych strojov pomocou grafického rozhrania, kde používateľ systémom drag-and-drop vytvorí schému prostredia. Pomocou komponentu vRealize môže používateľ automatizovať nielen vlastnú infraštruktúru postavenú na produktoch spoločnosti VMware, ale taktiež aj prostredia vo verejnom CC spoločnosti Amazon – AWS [54].

Z vyššie uvedeného prehľadu privátnych CC platforiem IaaS vyplýva, že spoločnosti najviac využívajú vSphere od spoločnosti VMware, na druhom mieste sa striedavo umiestňovali VMware vCloud a Microsoft Azure Stack. V poslednom ročníku ankety sa na druhom mieste umiestnil OpenStack, ktorý má aj naďalej stúpajúcu tendenciu. Všetky vyššie uvedené CC platformy, či už verejné alebo privátne, ponúkajú službu IaaS.

4 Návrh všeobecného modelu popisu virtuálneho prostredia

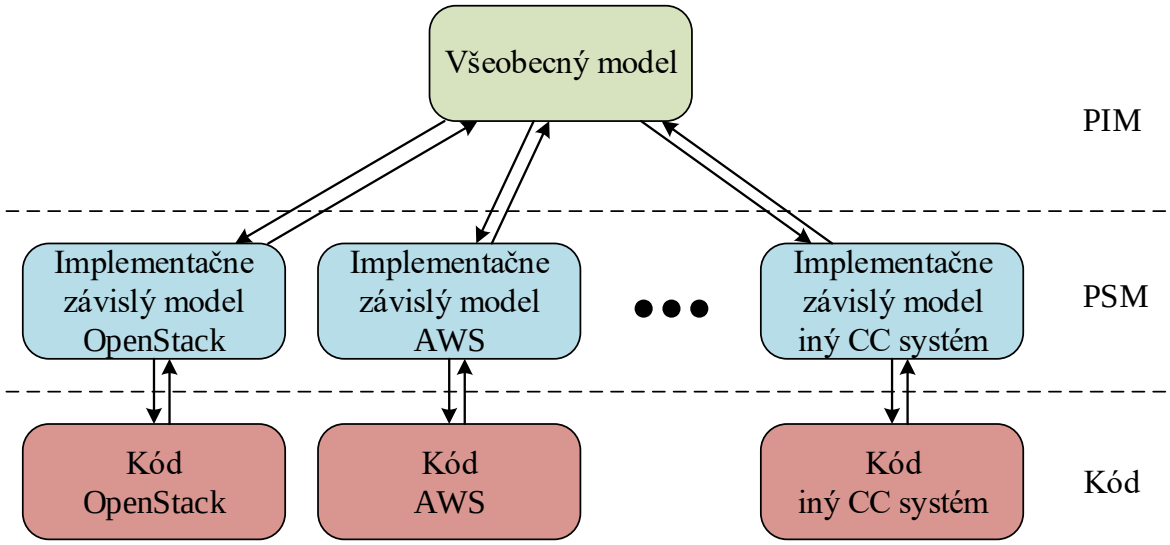
Pri návrhu všeobecného modelu popisu virtuálneho prostredia sme zobrali ako vzor v súčasnosti dominantných poskytovateľov verejného CC a to Amazon AWS, Microsoft Azure, a jedného z najrozšírenejších zástupcov privátneho CC, systém OpenStack. Tento výber vyplynul z vyššie uvedeného prehľadu a podľa prieskumu [13]. Tieto tri platformy sú globálne dominantnými poskytovateľmi CC IaaS prostredí vo firemnom sektore.

Cieľom práce je vytvorenie všeobecného, nezávislého popisu prostredia, ktorý bude využiteľný pre akéhokoľvek poskytovateľa CC. Pri návrhu a tvorbe všeobecného modelu sme použili modelom riadený prístup, presnejšie princíp modelom riadenej architektúry (MDA), ktorá sa používa pri vývoji softvéru. MDA sa skladá zo štyroch vrstiev – CIM, PIM, PSM a „kód“. Vrstva CIM (*Computer Independent Model*) sa používa pre biznis popis aplikácie alebo informačného systému. Sú v nej uvedené požiadavky na systém ako celok a na funkcie jednotlivých častí systému. V našom návrhu od vrstvy CIM úplne abstrahujeme, pretože budeme riešiť transformáciu už funkčného popisu prostredia, ktorý už vznikol z určitých požiadaviek, a tieto požiadavky sa počas transformácie nebudú meniť. CIM vrstva by tak bola pri popise prostredia v akomkoľvek funkčnom jazyku totožná.

Taktiež abstrahujeme od vrstvy „kód“. V našom ponímaní je táto vrstva implementovaná u každého poskytovateľa CC individuálne. Kód vnímame ako inštrukcie riadiacemu prvku celého CC prostredia, ktoré mu odovzdá entita parsujúca špecializovaný popis prostredia. Inými slovami, ak vytvoríme popis prostredia pre konkrétneho poskytovateľa CC (vrstva PSM), jeho implementačné prostredie musí byť schopné tento popis spracovať a následne odovzdať inštrukcie riadiacemu softvéru, nazývanému hypervízor, ktorý sa postará o vykonanie a vytvorenie prostredia definovaného popisom. Takáto akcia sa ale musí udiať u každého poskytovateľa CC osobitne a špecializovane vzhľadom na poskytovateľa, pretože implementácia hypervízora môže byť u každého poskytovateľa špecifická. Z tohto dôvodu bude nami navrhovaný model implementačne nezávislý (vrstva PIM).

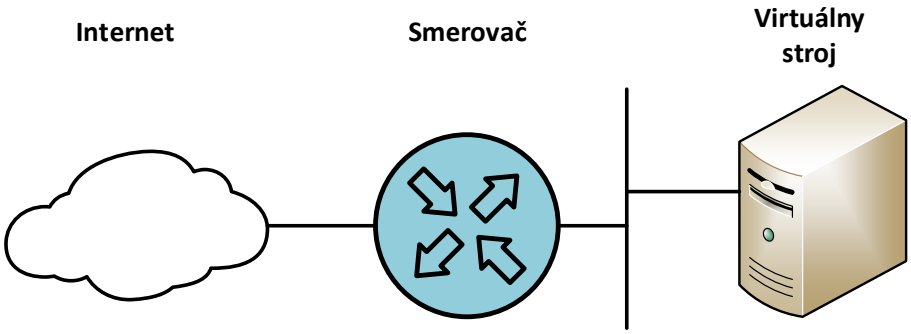
Na obrázku Obrázok 12 je znázornený koncept navrhovaného modelu v rámci paradigmy MDA. Každý CC poskytovateľ ponúka možnosť svojim používateľom vytvoriť popis prostredí v jazyku špecifickom pre svoje prostredie. Tento popis radíme do roviny PSM. Riešený model radíme do roviny PIM, pretože nie je zložený zo špecifických

komponentov určitej CC implementácie, no zachováva všetky atribúty a vlastnosti prostredia, z ktorých by sme mohli vytvoriť funkčný platformovo závislý popis.



Obrázok 12 Návrh všeobecného modelu popisu virtuálneho prostredia, zdroj autor

Virtuálna topológia sa môže skladať z desiatok logických entít IaaS služby, ktoré sú navzájom prepojené. Pre účely riešenia tejto práce nie je potrebné vytvoriť všeobecný model pre všetky z nich, ale navrhnúť základný model len pre niektoré entity, ktorý je možné v budúcnosti rozvíjať. Pre demonštračné a testovacie účely sme preto navrhli jednoduchú vzorovú topológiu IaaS služby, ktorá sa skladá z jedného virtuálneho stroja (VM) v privátnej sieti za logickým smerovačom CC prostredia. Vychádzame z vlastnej skúsenosti a taktiež z domnienky, že takáto topológia býva základom aj pre rozsiahlejšie topológie virtuálnych prostredí riešených ako IaaS CC služby. Diagram takejto topológie môžeme vidieť na obrázku Obrázok 13.



Obrázok 13 Testovacia topológia, zdroj autor

V časti 4.1 sú analyzované a uvedené popisy základných logických entít, pomocou ktorých je možné vytvoriť danú vzorovú topológiu. Ďalej sa tu nachádza analýza týchto entít v konkrétnych CC prostrediach, návrh entít vo všeobecnom modeli a následne návrh transformačných pravidiel.

4.1 Identifikácia entít IaaS služby

Ako už bolo spomenuté, pri návrhu modelu sme vychádzali z najrozšírenejších platforiem na trhu – Amazon AWS, OpenStack a Microsoft Azure. Vo všetkých týchto platformách je možné vytvoriť popis prostredia pomocou deklaratívneho jazyka, ktorý popisuje parametre prostredia, jednotlivé entity, ich atribúty a vzájomné prepojenia a závislosti týchto entít. Tento koncept sme zachovali aj pri návrhu, a navrhovaný model je taktiež rozdelený na dve hlavné časti – parametre a zdroje (*resources*).

Parametre prostredia slúžia na definovanie hodnôt administrátorom, ktoré je možné jednoducho meniť a nachádzajú sa v popise viackrát. Administrátor takto môže zmenou jedného parametra ovplyvniť viac výskytov jednej hodnoty. Ako príklad môžeme uviesť obraz virtuálneho stroja, z ktorého sa budú vytvárané virtuálne stroje klonovať. Ak sa v topológii nachádza niekoľko identických serverov, v prípade že administrátor požaduje zmenu obrazu pre všetky výskyty, stačí zmeniť jednu hodnotu. Pri každej definícii virtuálneho stoja je definovaný odkaz na parameter, odkiaľ má získať hodnotu. Parametrizovať je možné rôzne hodnoty, či už spomínané obrazy virtuálnych strojov, rozsahy IP adries, názvy virtuálnych sietí či rozsah blokovaných portov pre firewall.

V časti zdrojov sú definované vlastné entity, ktoré sa budú v topológii vyskytovať. Tu prebieha samotná konfigurácia prostredia, definovanie atribútov, závislostí a prepojení

medzi entitami. Pre náš návrh a overenie sme vybrali tieto základné entity: sieť, podsieť, virtuálny stroj a bezpečnostná skupina, pretože tvoria základ každej IaaS topológie.

Pre vytvorenie jednoduchej topológie, ako je znázornení na obrázku Obrázok 13 sme použili päť vybraných základných entít, ktoré sú nevyhnutné pre jej funkčnosť. Popisy týchto entít, ako aj ich funkcie vo virtuálnych prostrediach sú uvedené nižšie.

4.1.1 Parametre prostredia

Väčšina popisných skriptov obsahuje parametre. Sú to hodnoty, ktoré definuje administrátor a ktoré mu uľahčujú prácu so skriptom popisujúcim virtuálne prostredie (vrstva PSM). Hodnoty týchto parametrov sú zvyčajne čísla a textové reťazce, no môžu nadobúdať aj hodnoty pravdivostnej logiky.

Na parametre je možné odvolávať sa v rámci definície logických entít, ako napríklad virtuálnych strojov, sietí, bezpečnostných skupín a podobne. Analógiu môžeme nájsť v definovaní konštanty v rôznych programovacích jazykoch. Ak programátor potrebuje zmeniť viac výskytov jednej hodnoty, stačí zmeniť definovanú hodnotu konštanty, čo má za následok zmenu hodnoty v celom programe, kde sa konštanta použila.

Každý parameter obsahuje hodnotu, no môže obsahovať aj rôzne obmedzenia hodnoty, ktoré môže administrátor zadať. Sú to napríklad regulárne výrazy ktorými sú obmedzené znaky hodnoty, zoznam či interval povolených hodnôt. Konkrétne typy obmedzení budú detailne popísané pri implementáciách CC prostredí.

4.1.2 Virtuálny stroj

Virtuálny stroj (VM) je väčšinou kľúčovým prvkom topológie, preto je naň kladený dôraz pri popise, a taktiež je bohato konfigurovateľný v rámci popisu prostredia. V rôznych systémoch má rôzne parametre, no vo všetkých sú to najmä meno VM, obraz, z ktorého sa bude VM klonovať a určitá forma šablóny, na základe ktorej bude mať VM pridelené výpočtové prostriedky (CPU, RAM, HDD a pod.). Toto sú zvyčajne základné parametre, bez ktorých nie je možné VM v IaaS prostredí spustiť. Následne je možné VM prideliť rôzne dodatočné parametre, ako napríklad sieťové rozhrania, zabezpečenie na základe filtrovania paketov, pridelenie SSH kľúča, či dátové centrum, prípadne konkrétny server,

kde by sa mala VM vytvoriť. Viac sú tieto parametre popísané nižšie, už pri konkrétnych implementáciách CC prostredí.

4.1.3 SSH kľúč

Ako napovedá názov, SSH kľúč slúži na prístup k VM pomocou protokolu SSH (*Secure Shell*). Tento protokol bol vyvinutý pre UNIX-ové systémy, no dnes ho je možné používať aj na pripájanie k iným OS, ako napríklad MS Windows. SSH poskytuje šifrovaný prístup k VM na báze príkazového riadku, cez ktorý je možné systém plnohodnotne ovládať. V CC prostrediach je SSH jedným zo základných protokolov na vzdialený prístup a ovládanie VM. Protokol SSH používa algoritmy kryptografie verejného kľúča (*PKI – Public Key Infrastructure*), kde je potrebný pár kľúčov. Verejný a privátny kľúč, ktoré tvoria neoddeliteľnú dvojicu. Aby bolo možné kľúče použiť, jeden z nich (zvyčajne verejný) musí byť nahratý na VM a druhý použije administrátor pre získanie prístupu k serveru. Entita SSH kľúča slúži na vygenerovanie dvojice kľúčov a importovanie verejného kľúča do VM, aby bolo možné pripojiť sa na ňu.

Pri vytváraní kľúča je potrebné zadať len meno kľúča, ktoré slúži ako jedinečný identifikátor. Voliteľné parametre sú *uloženie kľúča*, *vloženie verejného kľúča* a *používateľ*.

Parameter *uloženie kľúča* hovorí, či administrátor chce dvojicu kľúčov uložiť a v budúcnosti ešte tento pár použiť pri inej VM. Ak nie, vygenerovaný kľuč sa importuje do VM a zahodí, čím vznikne kľúč na jedno použitie, pri zmazaní VM bude zmazaná aj jedna polovica kľúčového páru, čím sa druhá polovica stane nepoužiteľnou.

Druhý voliteľný parameter *vloženie verejného kľúča* určuje, či chce administrátor VM použiť už existujúci verejný kľúč pre prístup k VM. Ak áno, nebude generovaný nový pár kľúčov, ale zadaný kľúč sa nahrá do VM. Administrátor VM tým vytvorí situáciu, kde sa môže pomocou jedného privátneho kľúča pripájať k viacerým nezávislým inštanciám VM.

Posledný parameter je *používateľ*, pomocou ktorého je možné vygenerovaný kľúč priradiť konkrétnemu používateľovi. Takéto priradenie sa deje z bezpečnostného hľadiska, kde sa kľúč priradí jednému používateľovi, a ostatní používatelia v rámci CC prostredia nemajú ku kľúčovému páru prístup. V prípade, ak by bol kľúč zdieľaný, ostatní používatelia by mali prístup len k verejnému kľúču, cez ktorý by potencionálne mohli vypočítať druhú polovicu kľúča, čím by dokázali ovládať VM iných používateľov.

4.1.4 Bezpečnostná skupina

Bezpečnostná skupina (*Security Group*) funguje ako jednoduchý filter paketov. Každá VM má pridelenú aspoň jednu bezpečnostnú skupinu, ktorej pravidlá sa aplikujú na každý paket, ktorý prichádza alebo odchádza z VM. Každá bezpečnostná skupina obsahuje pravidlá, ktoré sú vyhodnocované sekvenčne. V CC prostrediach ktoré sme analyzovali, dokážu tieto pravidlá len zakazovať pakety. Ak niektorý paket vyhovuje porovnávaciemu pravidlu je zahodený. Zvyčajne bez ICMP správy, ktorá informuje odosielateľa o zahodení paketu. Ak paket nesplní ani jedno pravidlo, je prepustený ďalej. V pravidlách je možné len zakázať prechod paketu, na konci každej skupiny je implicitné povolenie prechodu. V prípade, že je na jednu VM aplikovaných niekoľko bezpečnostných skupín, pravidlá v nich obsiahnuté vytvoria logický súčet (*or*) a paket je porovnávaný s každým pravidlom. V pravidlách je typicky možné porovnávať hodnoty len v hlavičkách sieťovej a transportnej vrstvy ISO OSI modelu.

4.1.5 Sieť

V CC IaaS systémoch je entita „sieť“ chápaná ako kontajner pre podsiete. Väčšinou nemá vlastný adresný rozsah a slúži čisto ako logický identifikátor. Typicky sa jedna takáto entita pridelí jednému používateľovi, v ktorej si používateľ vytvára podsiete so špecifickými parametrami (adresa siete, adresa DNS servera, podpora DHCP a pod.).

Každá sieť obsahuje unikátny identifikátor, na základe ktorého sú identifikované aj všetky jej podsiete. Na základe tohto identifikátora potom môže poskytovateľ CC prostredia tarifikovať jednotlivých používateľov, napríklad za objem prenesených dát. Okrem identifikácie z pohľadu manažmentu sú identifikátory využívané aj na sieti, kde môžu byť reštrikcie pre používateľa na základe identifikátora. V CC prostrediach sa využívajú virtualizačné techniky siete, ako napr. VXLAN alebo NVGRE, ktoré vytvárajú tunely medzi rôznymi prvkami siete. Na identifikátory tunelov sa veľmi často používajú práve identifikátory sietí, a z toho je možné odvodiť, ktorý používateľ používa ktoré tunely.

Sieť nemá veľa parametrov. Zvyčajne je to len meno, prípadne CIDR (*Classless Inter-Domain Routing*) zápis rozsahu sietí, ktoré sa môžu použiť v jej podsieťach. Amazon AWS používa CIDR zápis, a podsiete môžu mať rozsahy len z tohto rozsahu. Na druhej

strane v systéme OpenStack sa vôbec nepoužíva CIDR zápis a sieťové adresy podsietí sú úplne nezávislé od nadradenej siete.

4.1.6 Podsieť

Entita podsiete vždy patrí do nejakej siete. Sieť je v podstate kontajner, ktorý združuje niekoľko podsietí. Podsieť je už vlastnou sieťou, do ktorej sa pripájajú VM. Každá podsieť je definovaná prefixom, ktorý je jedinečný v rámci jednej siete. Ďalšie parametre, ktoré sa definujú v rámci podsiete často bývajú adresa brány, adresa DNS servera, verzia IP protokolu či meno, ktoré má len informačný charakter pre administrátorov. Podsiete sú navzájom nezávislé, a pre prechod dát medzi podsieťami je potrebné smerovanie.

Ako bolo spomenuté vyššie, vybrali sme len základné entity, pomocou ktorých vieme definovať vzorovú topológiu vytvorenú pre účely riešenia práce. Každý CC systém môže mať desiatky logických entít tvoriacich virtuálne prostredie. Ako príklad uvedieme, že systém OpenStack definuje okolo sto rôznych logických entít a systém AWS dokonca niečo cez dvestopäťdesiat.

4.2 Entity služby v existujúcich IaaS riešeniach

Pre každé vybrané CC riešenie sú entity vybraných služieb rôzne riešené. Ich popis je v nasledujúcich podkapitolách.

4.2.1 Amazon Web Service

V nasledujúcej časti je prehľad vybratých komponentov vzorovej IaaS služby tak, ako ich definuje a používa AWS.

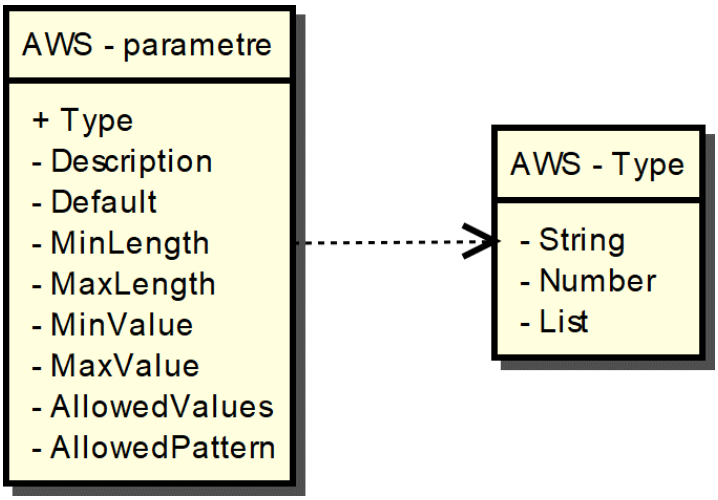
Parametre prostredia

Systém AWS ponúka využitie parametrov v popisných skriptoch, kde v jednom skripte môže byť najviac šesťdesiat parametrov. Každá entita *parametre prostredia* má vlastné parametre, kde jediným povinným parametrom je *Type*. Ten hovorí o type hodnoty, ktorú môže nadobudnúť. V systéme AWS to môže byť reťazec, číslo a zoznam reťazcov oddelených čiarkou. Na základe typu je potom možné definovať obmedzenia hodnoty, ktoré môže entita parametra nadobúdať. Z nepovinných parametrov je to popis logickej entity daného parametra a implicitná hodnota. Implicitná hodnota môže byť predefinovaná

pri spustení skriptu, no ak administrátor neurčí inak, použije sa práve definovaná implicitná hodnota.

Ďalšie nepovinné parametre obmedzujú použiteľné hodnoty v entite parametra. Pri reťazci to môže byť minimálna alebo maximálna dĺžka, prípadne povolené znaky (*AllowedPattern*), ktoré sa definujú pomocou regulárnych výrazov. Pri číselných hodnotách je možné obmedziť interval, z ktorého môžu hodnoty pochádzať. Čísla môžu nadobúdať ako celé, tak aj desatinné hodnoty. Posledným parametrom je *AllowedVallues*, ktorý obsahuje zoznam predvolených hodnôt. Z tých si môže administrátor jednu vybrať pri aplikovaní skriptu. Benefit je v tom, že nemôže dôjsť k preklepu, prípadne k syntaktickej chybe.

Na obrázku Obrázok 14 sú znázornené parametre, ktoré môžu entity parametrov obsahovať. Kvôli názornosti je vyčlenený typ, pretože na základe jeho hodnoty môžu byť niektoré parametre ignorované.



Obrázok 14 Parametre prostredia v systéme AWS, zdroj autor

Virtuálny stroj

Virtuálny stroj (*VM – Virtual Machine*) je jedným zo základných prvkov tvorby popisných skriptov. V skriptoch sa označuje ako *AWS::EC2::Instance*, a je bohato konfigurovateľný. VM má v systéme AWS jeden povinný parameter. Je ním *ImageId*, čo je referencia na obraz operačného systému, z ktorého bude daná VM klonovaná. Je možné použiť obrazy ktoré nachystal poskytovateľ AWS, alebo je možné si vytvoriť

vlastné obrazy a tie importovať do systému AWS. V každom prípade už musia existovať, aby bolo možné vytvoriť klonovanú VM.

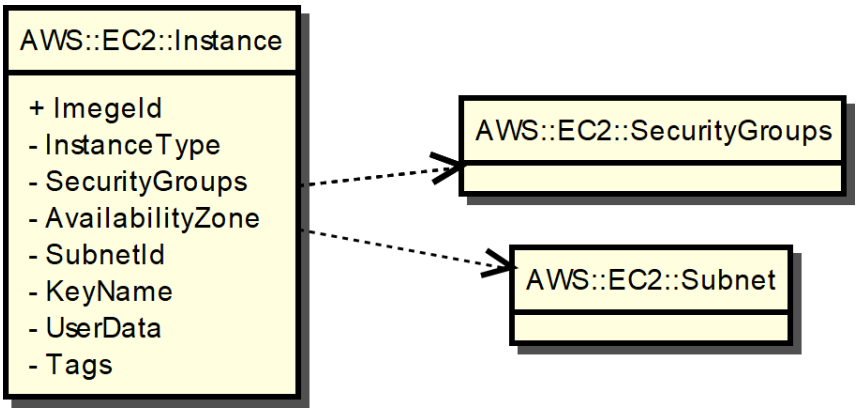
Všetky ostatné parametre sú nepovinné. Parameter *InstanceType* je šablóna, podľa ktorej budú VM pridelené výpočtové prostriedky, ako napríklad počet jadier CPU, pamäť RAM, či pevný disk. Na základe využívania týchto hodnôt je následne používateľ spoplatňovaný. Šablóny nie je možné vytvárať na požiadanie, teda je možné použiť len existujúce, ktoré ponúka AWS. Pokiaľ tento parameter nie je definovaný, použije sa predvolená hodnota. Ďalším parametrom sú bezpečnostné skupiny, ktoré fungujú ako paketové filtre. Viac o bezpečnostných skupinách sa nachádza v časti 4.4.4 Bezpečnostná skupina.

Parameter *AvailabilityZone* označuje dátové centrum spoločnosti Amazon, kde sa daná VM vytvorí. V súčasnosti má spoločnosť Amazon 16 dátových centier umiestnených po celom svete. Takto si môže používateľ vybrať, kde sa nová VM vytvorí. To môže mať dôležitý dopad na jeho dáta, pretože rôzne krajiny majú rôznu legislatívu na prístup k privátnym dátam. Ďalším parametrom je *SubnetId*, čo je referencia na podsieť, do ktorej bude VM pripojená. Týchto parametrov môže byť viac, čím administrátor docieli pripojenie jednej MV do viacerých podsietí. Viac o podiseťach sa nachádza v časti 4.4.6 Podsieť Parameter *KeyName* odkazuje na SSH kľúč, ktorý sa použije na vzdialený prístup k VM. SSH kľúčom sa venujeme v časti 4.4.3 SSH kľúč.

Veľmi dôležitým parametrom je *UserData*. Pomocou tohto parametra je možné vo VM spustiť skript pri prvom bootovaní. Vlastný skript môže byť v akomkoľvek jazyku, ktorý je podporovaný operačným systémom, či už je to UNIX-ový Shell, alebo PowerShell pre systémy Microsoft Windows. Na základe týchto skriptov je možné modifikovať VM podľa požiadaviek. Či už sú to rôzne nastavenia alebo inštalácie programov, po použití takéhoto skriptu sa vytvorí VM, ktorú nie je potrebné ďalej modifikovať a môže ihneď slúžiť zámeru, s ktorým bola vytváraná. Posledným parametrom sú tagy, čo sú len jednoduché prívlastky, ktoré slúžia administrátorom pre jednoduchšiu identifikáciu a utrieďovanie virtuálnych inštancií.

Na obrázku Obrázok 15 sa nachádza diagram závislostí pre VM v CC systéme AWS. Parametre označené symbolom „plus“ sú povinné parametre, parametre označené symbolom „mínus“ sú nepovinné parametre. Môžeme vidieť, že VM nemá žiadne závislosti a môže

existovať úplne samostatne. Obidve entity, ku ktorým ukazuje šípka sú nepovinné a tento vzťah len naznačuje referenciu na danú logickú entitu.



Obrázok 15 VM v systéme AWS, zdroj autor

SSH kľúč

V systéme AWS nie je možné vytvoriť SSH kľúč pomocou popisného skriptu. Jediná možnosť vytvorenia kľúča je cez webovú stránku nazývanú „*AWS Management Console*“. Tam si používateľ vytvorí a pomenuje kľúč, prípadne uloží jeho privátnu časť. V popisných skriptoch sa potom už len odvolá na meno existujúceho kľúča, ktorý bude importovaný do vytváranej VM. Používatelia tak nemajú možnosť vytvorenia SSH kľúčov v skriptoch a musia si kľúče vytvoriť predtým, ako začnú vytvárať popisné skripty prostredia.

Bezpečnostná skupina

Bezpečnostná skupina môže existovať úplne autonómne. Jej názov je „*AWS::EC2::SecurityGroup*“ a je ju možné definovať cez webové rozhranie, ako aj cez popisný skript. V systéme AWS má každá bezpečnostná skupina jeden povinný parameter, popis danej skupiny. Meno skupiny je jedinečný identifikátor v rámci jedného projektu, ktorý ale nie je povinným parametrom. Ak meno bezpečnostnej skupiny nie je definované, systém AWS vygeneruje unikátne meno pre danú skupinu. Nasledujú ďalšie dva nepovinné parametre, referencia na skupiny vstupných a výstupných pravidiel. Posledným parametrom sú *Tagy,* ktoré slúžia opäť len na označenie a popis skupiny.

Skupiny vstupných a výstupných parametrov majú totožnú syntax. No ako napovedá názov, líšia sa len v smere nasadenia filtrovania paketov. Smer sa určuje z pohľadu VM,

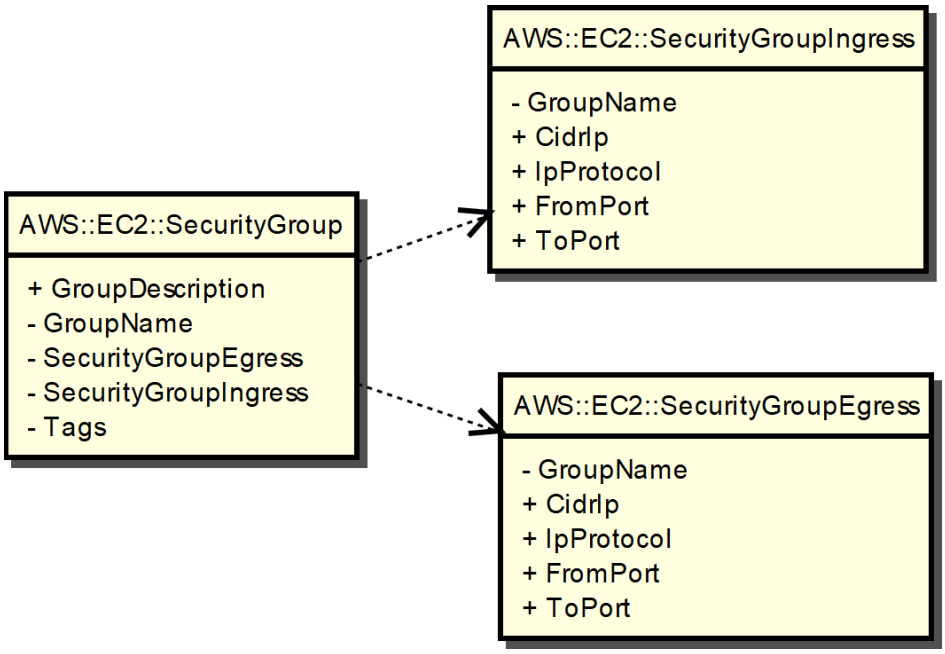
to znamená, že vstupné pravidlá budú aplikované na pakety, ktoré sú určené pre danú VM, a výstupné na tie, ktoré vygenerovala VM. Tieto skupiny môžu byť zdieľané pre viaceré bezpečnostné skupiny, nakoľko to sú samostatné logické entity, ktoré majú názov „*AWS::EC2::SecurityGroupIngress*“, resp. „*AWS::EC2::SecurityGroupEgress*“.

V týchto skupinách sa nachádzajú štyri povinné parametre. Prvým z nich je *IpProtocol*, ktorý trochu zavádzajúco označuje transportný protokol nesený v pakete. Môže nadobúdať číselné alebo slovné parametre. Zo slovných je to *TCP*, *UDP* a *ICMP*. Číselné hodnoty označujú čísla príslušných protokolov podľa IANA (*Internet Assigned Numbers Authority*), napríklad *TCP* má číselnú hodnotu „6“. Ak by administrátor chcel špecifikovať všetky transportné protokoly, nastaví hodnotu parametra na „-1“.

Ďalšie dva povinné parametre spolu súvisia. Sú to *FromPort* a *ToPort*. Tieto dva parametre určujú interval portov, ktorý bude porovnávaný v danom pravidle. Ak by administrátor potreboval definovať len jeden port, obidva tieto parametre nastaví na rovnakú hodnotu. V prípade transportného protokolu ktorý nepoužíva transportné porty (napr. ICMP), sú tieto hodnoty ignorované, no z hľadiska zrozumiteľnosti a čitateľnosti skriptu sa odporúča zadať hodnotu, ktorá je neprípustná, napríklad „-1“.

Posledným parametrom je *CidrIp*, ktorý určuje rozsah IP adries, ktoré budú v danom pravidle porovnávané. Ako napovedá názov, hodnota sa zapisuje v *CIDR* formáte. Tento parameter má rozdielne vnímanie vo vstupných a výstupných pravidlách. Pri vstupných pravidlách sa porovnáva so zdrojovými adresami, zatiaľ čo pri výstupných pravidlách s cieľovými IP adresami. Pokiaľ by administrátor nastavil hodnotu na „0.0.0.0/0“, označil by tým úplne všetky adresy, čo by malo za následok porovnávanie všetkých paketov s týmto pravidlom.

Z nepovinných parametrov tu môžeme nájsť názov skupiny, ktorý je opäť jedinečný v rámci jedného projektu. Diagram závislostí bezpečnostnej skupiny v CC systéme AWS je znázornený na obrázku Obrázok 16.

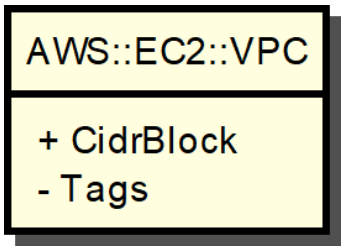


Obrázok 16 Bezpečnostná skupina v systéme AWS, zdroj autor

Sieť

V systéme AWS má sieť jediný povinný parameter – „CidrBlock“, ktorý obsahuje CIDR záznam. Všetky podsiete z tejto siete musia mať adresy daného CIDR záznamu. Ďalej obsahuje nepovinné atribúty ako napríklad *tagy*. O sieť sa stará modul EC2, a v skriptoch sa zapisuje ako AWS::EC2::VPC, kde skratka VPC znamená „Virtual Private Cloud“.

Diagram závislostí sa nachádza na obrázku Obrázok 17. Parametre označené symbolom plus sú povinné, parametre označené symbolom mínus sú nepovinné.

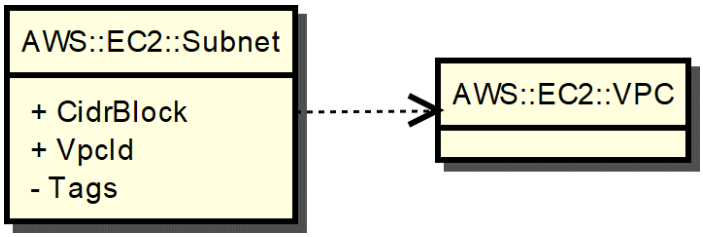


Obrázok 17 Sieť v systéme AWS, zdroj autor

Podsieť

V systéme AWS sú podsiete súčasťou entity sieť. Entita siete má ako povinný atribút aj IP prefix, a prefixy podsietí musia byť z rozsahu, ktorý je definovaný v sieti. Inými slovami, entita siete sumarizuje všetky prefixy zo svojich podsietí. Ďalší povinný atribút je referencia na entitu siete. Pomocou tejto referencie je podsieť jednoznačne pridelená k jednej sieti, čo má za následok možnosť vytvárať podsiete s rovnakými prefixami pre rôznych používateľov. Ako bolo spomenuté vyššie, entita siete má jednoznačný identifikátor, ktorý týmto prechádza aj na všetky jej podsiete. Spoločnosť Amazon tak môže jednoznačne monitorovať a prípadne spoplatňovať svojich používateľov na základe tohto identifikátora.

Na obrázku Obrázok 18 je znázornený model závislostí podsiete v systéme AWS. Povinné parametre sú označené symbolom plus. Parameter *VpcId* odkazuje na entitu siete, ku ktorej daná podsieť patrí. Parameter *CidrBlock* je IP prefix, ktorý musí byť podsieťou z prefixu, ktorý je definovaný v sieti, na ktorú odkazuje *VpcId*.



Obrázok 18 Podsieť v systéme AWS, zdroj autor

Niektoré hodnoty, ktoré je možné konfigurovať v iných CC systémoch, sú určené poskytovateľom CC prostredia AWS. Napríklad adresy DNS serverov sú vopred zadané, a pre VM v tomto prostredí je možné používať výhradne DNS servery poskytované spoločnosťou Amazon. Taktiež nie je možné nastaviť IP adresu brány v danej podsieti. Tá bola opäť stanovaná poskytovateľom na pevnú hodnotu, ktorú používateľ ani nepozná. Všetky VM v tomto systéme získavajú IP adresy pomocou DHCP protokolu, cez ktorý získajú aj adresu svojej brány, ktorá sa môže týmto meniť bez toho, aby bola ovplyvnená dátová prevádzka.

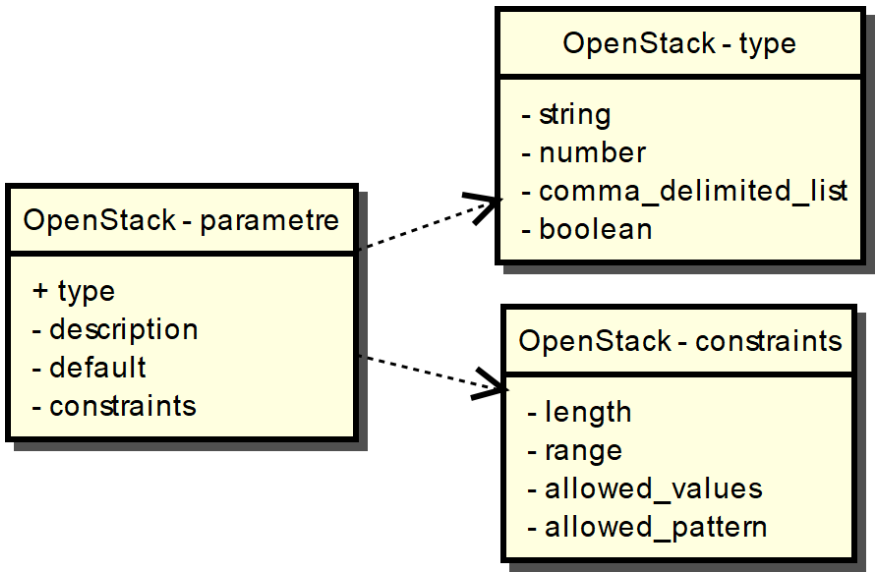
4.2.2 OpenStack

V nasledujúcej časti sa nachádza prehľad identifikovaných IaaS entít spolu s popismi povinných a nepovinných parametrov v tomto systéme.

Parametre prostredia

Podobne ako AWS, aj systém OpenStack povoľuje používanie parametrov v popisných skriptoch. Taktiež entity parametrov majú vlastné parametre, kde povinným je typ. Typ určuje hodnotu parametra, ktorými môže byť reťazec, číslo, zoznam reťazcov a hodnota Boolovej algebry. Na rozdiel od systému AWS OpenStack dovoľuje použitie logických výrazov pravda/nepravda, na základe ktorých je potom možné pri definovaní logických entít vetviť program a tak lepšie prispôsobiť výsledné riešenie.

Z nepovinných parametrov to je popis daného parametra, implicitná hodnota a obmedzenia vlastnej hodnoty parametra. Implicitná hodnota je použitá v prípade, ak administrátor pri aplikovaní skriptu nepredefinuje hodnotu parametra. *Constraints* obsahuje zoznam obmedzení pre vlastnú hodnotu parametra. Podobne ako v systéme AWS môže administrátor definovať minimálnu a maximálnu dĺžku reťazca, interval hodnôt pre číselné parametre, či regulárny výraz obmedzujúci znaky v textovom reťazci. Nechýba ani zoznam hodnôt, z ktorých má administrátor možnosť výberu pri aplikovaní skriptu. Model parametra prostredia je na obrázku Obrázok 19.



Obrázok 19 Parametre prostredia v systéme OpenStack, zdroj autor

Virtuálny stroj

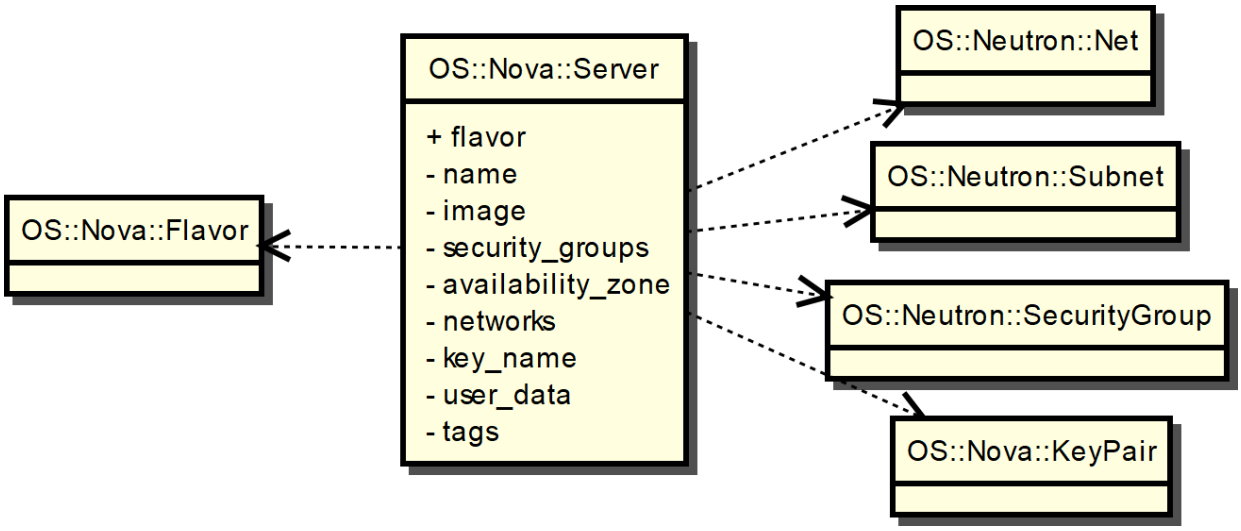
V popisných skriptoch sa VM v systéme OpenStack označuje ako *OS::Nova::Server*. Tento názov naznačuje, že o VM sa stará modul *Nova*, ktorý je všeobecne zodpovedný za vytváranie, prevádzku a modifikáciu virtuálnych inštancií koncových staníc. V systéme OpenStack má VM jeden povinný parameter. Na rozdiel od systému AWS to nie je obraz z ktorého sa bude nová VM klonovať, ale je to šablóna, podľa ktorej budú VM pridelené výpočtové zdroje (CPU, RAM, HDD). V systéme OpenStack sa táto šablóna nazýva *flavor*. Z nepovinných parametrov to je meno VM, ktoré je jedinečné v rámci projektu a názov obrazu operačného systému, z ktorého sa nová VM bude klonovať. OpenStack býva nasadzovaný väčšinou ako privátny CC systém, preto sa o obrazy (nazývané *image*) musí starať prevádzkovateľ systému. Existujú dve možnosti – buď si prevádzkovateľ bude vytvárať a aktualizovať vlastné obrazy, alebo si zadováži už existujúce obrazy. Väčšina Linuxových distribúcií ponúka zdarma svoje aktuálne obrazy systémov, ktoré stačí jednoducho importovať do CC systému.

Ďalší nepovinný parameter je bezpečnostná skupina (*SecurityGroup*), čo je jednoduchý filter paketov na úrovni CC systému. Viac o bezpečnostných skupinách sa nachádza v časti 4.4.4. Bezpečnostná skupina. Parameter *networks* slúži na pripojenie VM k sieti. V rámci tohto parametra je možné pripojiť VM do jednotlivej podsiete, alebo do všetkých podsietí v rámci jednej siete.

Ďalším nepovinným parametrom je referencia na SSH kľúč, ktorý slúži na vzdialený prístup a manažment VM. Parameter *user_data* dovoľuje v novovytvorenej VM spustiť ľubovoľný skript. Tento sa spustí len pri prvom spustení VM a dokáže modifikovať VM podľa predstáv administrátora, či už v rôznych nastaveniach systému, alebo môže nainštalovať a nastaviť nové aplikácie podľa potreby. Skript môže byť napísaný v akomkoľvek jazyku, ktorý je spustiteľný v danom operačnom systéme. Posledný nepovinný parameter je *tags*, čo sú pomocné označenia VM. Tie slúžia len pre administrátora, na základe ktorých môže VM triediť a filtrovať v rôznych výpisoch. Tento parameter nemá žiadny vplyv na funkčnosť VM a systému ako celku.

Na obrázku Obrázok 20 je znázornený diagram závislostí entity VM. Jediná závislosť je referencia na šablónu, podľa ktorej VM dostane pridelené výpočtové prostriedky. Táto šablóna môže už existovať a referencia bude odkazovať na meno existujúcej šablóny, alebo je možné šablónu priamo definovať v popisnom skripte. Na rozdiel od systému AWS si môžu používatelia definovať šablóny podľa vlastných potrieb, ak im existujúce šablóny

nevyhovujú. Na obrázku Obrázok 20 sa nachádzajú aj referencie na sieť, podsieť, bezpečnostnú skupinu a SSH kľúč. To sú nepovinné parametre, no vo viacerých scenároch sa VM na tieto entity odkazujú.



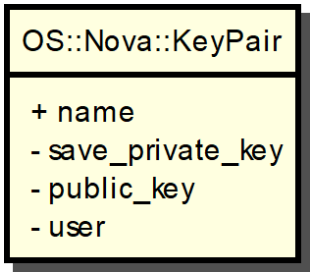
Obrázok 20 VM v systéme OpenStack, zdroj autor

SSH kľúč

V systéme OpenStack je možné kľúč vytvoriť aj cez webový prehliadač, prípadne cez príkazový riadok, aj priamo v rámci skriptu. Ak sa kľúč vytvára cez prehliadač, obdobne ako v systéme AWS kľúč musí existovať pred spustením skriptu a v rámci skriptu sa nachádza už iba referencia na meno kľúča, ktoré musí byť jedinečné.

Pri vytváraní kľúča v rámci popisného skriptu je postup veľmi podobný. Treba definovať meno, prípadne iné voliteľné parametre. Následne sa v rámci definovania VM nachádza referencia na vytvorený kľúč. O prístup k VM a manažment SSH kľúčov sa stará výpočtový modul Nova a entita má v skripte názov *OS::Nova::KeyPair*. Ak chce administrátor kľúč nielen uložiť do CC prostredia, ale aj ho zobraziť, je možné použiť výstupy, cez ktoré systém zobrazí privátnu časť dvojice kľúčov.

Diagram závislostí SSH kľúča sa nachádza na obrázku Obrázok 21. Kľúč nemá závislosti, môže existovať úplne samostatne nezávisle od VM, ktoré ho môžu následne použiť. Jediný povinný parameter je meno kľúča, ktoré musí byť jedinečné v rámci CC prostredia.



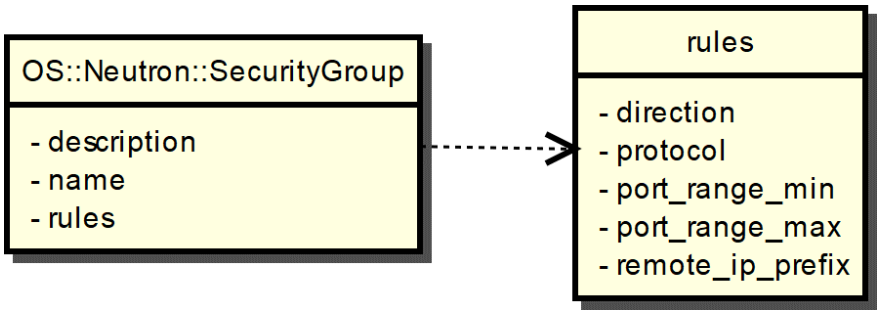
Obrázok 21 SSH kľúč v systéme OpenStack, zdroj autor

Bezpečnostná skupina

CC systém OpenStack používa obdobnú logiku bezpečnostnej skupiny ako systém AWS. Taktiež je to samostatná logická entita bez závislosti na iných entitách. Je ju možné definovať buď cez webové rozhranie, resp. cez príkazový riadok, alebo pomocou skriptu, kde jej názov znie „*OS::Neutron::SecurityGroup*“. Nemá žiadne povinné parametre, obsahuje len tri nepovinné. Prvým je *description*, ktorý slúži len ako popis danej skupiny. Druhým parametrom je jej meno, ktoré je jedinečné v rámci jedného projektu a tvorí akýsi identifikátor danej skupiny. Posledným parametrom je *rules*, čo sú vlastné bezpečnostné pravidlá. Na rozdiel od systému AWS bezpečnostné pravidlá nie sú ako samostatné logické entity. Pravidlá sú pevne integrované do bezpečnostnej skupiny. Na obrázku Obrázok 22, v diagrame závislostí sú pravidlá vyčlenené akoby samostatná entita, no je to len z dôvodu jednoduchšej orientácie a pochopenia, pretože každé pravidlo má vlastné parametre, ktoré definujú jednotlivé pravidlá.

Každé pravidlo má päť parametrov. Prvý je *direction*, ktorý označuje smer, či bude pravidlo aplikované na prichádzajúce, alebo na odchádzajúce pakety Pokiaľ nie je definovaný, implicitne sa porovnávajú prichádzajúce pakety. Ďalšie štyri parametre sú zhodné ako v systéme AWS. *Protocol* označuje transportný protokol nesený v tele paketu. Akceptované sú len tri hodnoty: *TCP, UDP* a *ICMP*. Číselné hodnoty nie sú prípustné. Ďalej dvojica parametrov *port_range_min* a *port_range_max* označujú interval portov, ktoré budú porovnávané v rámci pravidla. Posledným pravidlom je *remote_ip_prefix*, ktorý určuje rozsah IP adries porovnávaných v pravidle. Akceptovanou hodnotou je IP prefix v CIDR zápise.

Na obrázku Obrázok 22 je znázornený diagram závislostí bezpečnostnej skupiny v CC systéme OpenStack. Ako bolo spomenuté vyššie, skupina nemá žiadne závislosti a môže existovať ako nezávislá logická entita.

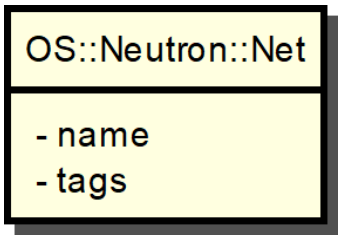


Obrázok 22 Bezpečnostná skupina v systéme OpenStack, zdroj autor

Sieť

V systéme OpenStack sa o siete stará modul Neutron. Ako vo všetkých moduloch, aj Neutron využíva pre názvy entít balíčkovaciu konvenciu. Sieť má názov „Net“ a zapisuje sa ako „*OS::Neutron::Net*“.

V tomto systéme sieť nemá žiadne povinné parametre, všetky parametre sú nepovinné. My sme zvolili pravdepodobne najpoužívanejšie parametre: meno a tagy. Model závislostí a používaní entity sieť v systéme OpenStack je znázornený na obrázku Obrázok 23.



Obrázok 23 Sieť v systéme OpenStack, zdroj autor

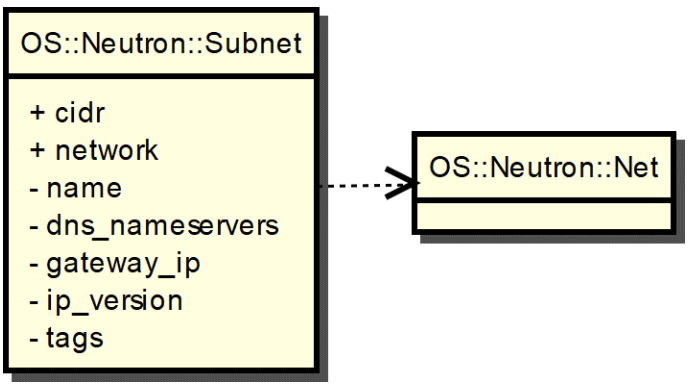
Podsieť

Oproti systému AWS je podsieť viac konfigurovateľná. Na jednej strane sú to dva zhodné povinné parametre (označené symbolom plus), na druhej strane sú tu nepovinné

parametre, ktoré AWS neponúka. Z povinných parametrov je to referencia na entitu siete, pod ktorú daná podsieť patrí, a zhodne ako v AWS, aj tu môže byť každá podsieť jednoznačne identifikovateľná na základe siete, pod ktorú patrí. Druhý parameter je IP prefix, ktorý môže byť ľubovoľný. Entita siete v systéme OpenStack nemá parameter IP prefixu a z toho vyplýva, že podsiete nemusia nutne byť z určitého pevne definovaného rozsahu.

Z nepovinných parametrov je to napríklad meno, ktoré má len informatívny charakter, ďalej adresu DNS servera, či adresu brány v danej podsieti. Na rozdiel od systému AWS je možné tieto adresy definovať pre každú podsieť zvlášť. Ďalším parametrom je verzia IP protokolu, kde môže administrátor nastaviť, či bude daná podsieť používať IPv4 alebo IPv6. Každá podsieť môže používať len jednu verziu súčasne. Ak by administrátor chcel zapojiť VM do siete IPv4 aj IPv6, musí vytvoriť dve podsiete, každú z inou verziou IP protokolu, VM prideliť dva sieťové rozhrania a následne každé rozhranie pripojiť do inej siete. V systéme OpenStack nie je možné prevádzkovať jednu podsieť v tzv. „*dualstack*“ režime. Posledným nepovinným parametrom sú *tagy*, čo sú informácie pre administrátorov, na základe ktorých môže podsiete triediť do rôznych kategórií bez ohľadu na ich funkčnosť alebo príslušnosť k nadradenej sieti.

Na obrázku Obrázok 24 je znázornený diagram závislostí. Jedinou entitou, ktorá musí existovať je sieť, pod ktorú bude podsieť patriť. Podsieť nemôže existovať samostatne, nezávisle od siete.



Obrázok 24 Podsieť v systéme OpenStack, zdroj autor

4.2.3 Microsoft Azure

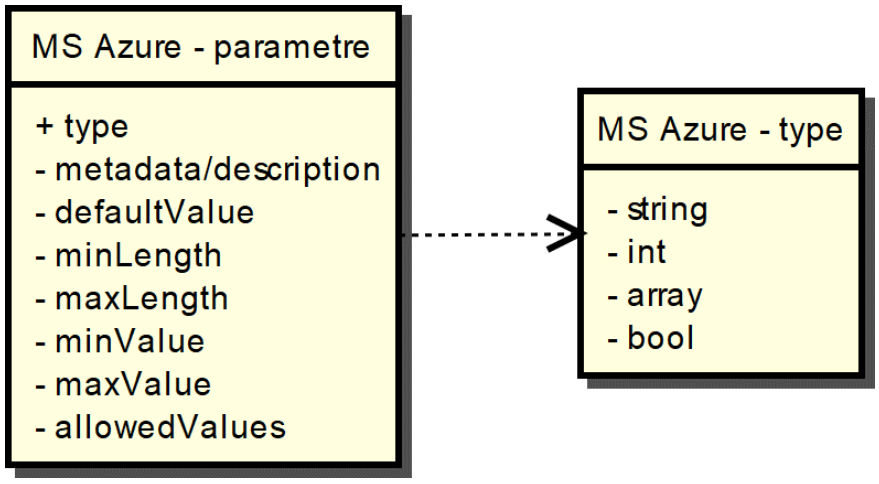
V nasledujúcej časti sa nachádzajú identifikované IaaS entity v tomto systéme, spolu s ich povinnými a nepovinnými parametrami.

Parametre prostredia

Rovnako ako v systémoch AWS a OpenStack, aj v systéme Azure je možné definovať parametre prostredia, na ktoré je možné odvolávať sa neskôr v popisných skriptoch virtuálnych prostredí. Aj v systéme Azure sa v entite *parametre prostredia* nachádza jediný povinný parameter – *type*. Ten definuje typ parametra, či je to textový reťazec, číslo, pole hodnôt, či logická hodnota typu pravda/nepravda.

Ostatné parametre sú nepovinné. Jedným z nich je popis daného parametra (*description*), ktorý je na rozdiel od ostatných parametrov o jednu úroveň hlbšie, v časti *metadata*. Medzi ďalšie parametre patria *implicitná hodnota*, ktorá je použitá v prípade, ak administrátor nezadá vlastnú hodnotu, ohraničenie dĺžky reťazca (*minLength* a *maxLength*), interval povolených hodnôt pre číselné hodnoty (*minValue* a *maxValue*), alebo zoznam hodnôt, z ktorých má administrátor možnosť výberu pri aplikovaní skriptu.

Na obrázku Obrázok 25 sú znázornené parametre entity *parametre prostredia*. Symbolom „plus“ je označený povinný parameter, a symbolom „mínus“ nepovinné parametre. Kvôli názornosti je vyčlenený typ, pretože na základe jeho hodnoty môžu byť niektoré parametre ignorované



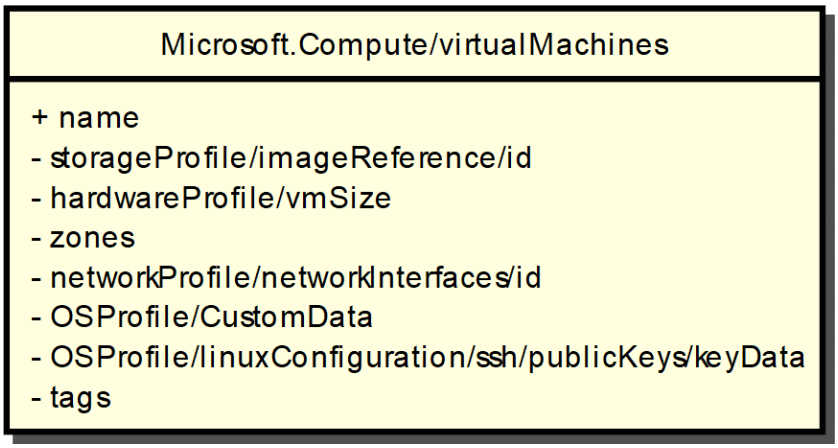
Obrázok 25 Parametre prostredia v systéme MS Azure, zdroj autor

Virtuálny stroj

Jednou zo základných entít je virtuálny stroj (VM). Microsoft Azure nepoužíva balíčkovaciu notáciu, ktorá oddeľuje jednotlivé časti hierarchie dvomi dvojbodkami. Namiesto dvoch dvojbodiek používa lomku. Virtuálny stroj sa nachádza v časti „Microsoft.Compute“, a celý názov definície VM je „Microsoft.Compute/virtualMachines“.

Podobne ako v systémoch AWS a OpenStack, aj v systéme Azure sa nachádzajú povinné a nepovinné parametre. Jediným povinným parametrom je názov VM - „name“. Všetky ostatné parametre sú nepovinné, a pokiaľ ich administrátor nenastaví v automatizačnom skripte, doplnia sa predvolenými hodnotami.

Na obrázku Obrázok 26 sú znázornené parametre VM v systéme Azure.



Obrázok 26 VM v systéme MS Azure, zdroj autor

Na rozdiel od systémov Amazon AWS a OpenStack sú parametre viac štruktúrované. Identifikátor obrazu, z ktorého sa bude VM klonovať má názov „*storageProfile/imgeReference/id*“, kde lomky oddeľujú jednotlivé úrovne. V tomto prípade sa samotné „id“ nachádza v časti „*imageReference“*, ktorá sa samotná nachádza v *„storageProfile*“. Šablóna pre VM, ktorá nastaví počet jadier CPU, veľkosť RAM a HDD má názov „*vmSize*“ a nachádza sa v časti „*hardwareProfile*“. Pri definícii VM je ju možné spustiť v bližšie špecifikovanej časti CC prostredia, ako napríklad definovať konkrétnu množinu, či jednotlivý fyzický server. Tieto servery sa nachádzajú v skupinách nazývaných zóny dostupnosti. Pre vytvorenie VM v špecifickej zóne je potrebné definovať direktívu „*zones*“.

V časti „*networkProfile/networkInterfaces/id*“ je možné definovať pripojenia VM do virtuálnych sietí. Vo väčšine CC IaaS prostredí je možné definovať inicializačný skript, ktorý sa vykoná pri prvom spustení VM. Aj v systéme Azure je možnosť spustiť takýto skript. Administrátor ho môže definovať v časti „*OSProfile/customData*“. Pre vzdialené prihlasovanie sa na VM je možné zadať hodnotu verejného kľúča. Tá sa definuje hlboko v štruktúre skriptu, konkrétne v „*OSProfile/linuxConfiguration/ssh/publicKeys/keyData*“. Posledným parametrom je „*tags*“, čo sú značky, ktoré môže administrátor priradiť konkrétnej VM. Použitie je identické ako v systémoch OpenStack a Amazon AWS.

SSH kľúč

V systéme Azure, podobne ako v systéme Amazon AWS nie je možné vytvoriť SSH kľúč pomocou automatizačného skriptu. Podľa dokumentácie je jediná možnosť vytvoriť kľúč vo webovom rozhraní a následne hodnotu verejného kľúča uviesť v skripte. Nie je možné odvolať sa na meno kľúča, vždy je nutné zadať hodnotu kľúča. Azure nepodporuje ani možnosť importovať do VM už existujúci SSH kľúč.

Bezpečnostná skupina

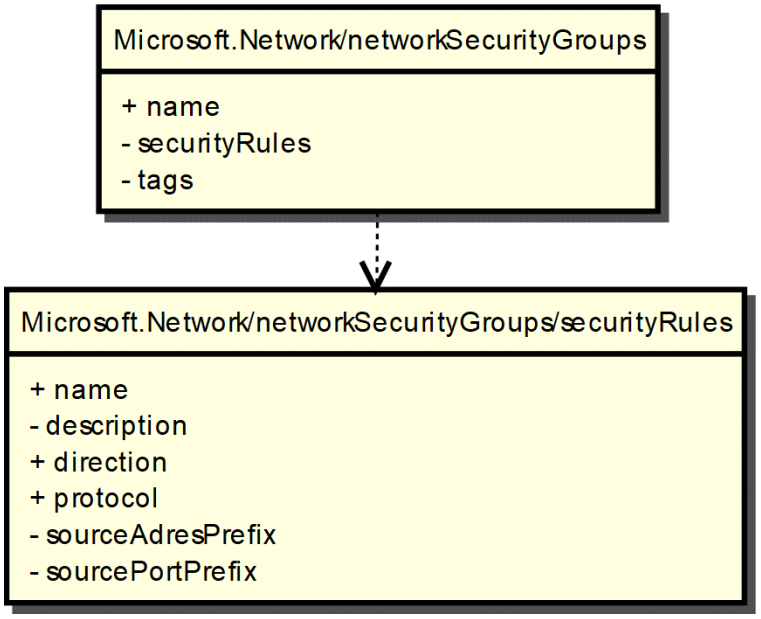
Bezpečnostná skupina funguje ako paketový filter, ktorým môže administrátor obmedzovať IP prevádzku VM na úrovni CC systému. Bezpečnostná skupina má rovnakú logickú štruktúru ako v systéme AWS, kde je definovaná entita skupiny, ktorá následne využíva ďalšie entity, v ktorých sú definované vlastné pravidlá. Entita bezpečnostnej skupiny sa definuje ako „*Microsoft.Network/networkSecurityGroups*“ a má jeden povinný parameter, „*name*“, ktorý označuje meno danej skupiny. Nasledujú dva nepovinné parametre, jedným z nich je „*securityRules*“, v ktorom sa nachádza zoznam entít pravidiel bezpečnostných skupín. Druhý nepovinný parameter je „*tags*“, ktorý dovoľuje administrátorom prideliť bezpečnostnej skupine určité značky.

Entita pravidiel bezpečnostných skupín je definovaná ako „*Microsoft.Network/networkSecurityGroups/securityRules*“. Má tri povinné a tri nepovinné parametre. Prvý povinný parameter je „*name*“, ktorý nastavuje meno pravidiel, ďalej to je „*direction*“, ktorý definuje smer toku paketov. Môže obsahovať dve hodnoty – ingress a egress, kde obe hodnoty sú definované z pohľadu VM. To znamená, že ak je nastavená hodnota ingress, porovnávané sú všetky pakety, ktorých cieľová adresa je adresa VM.

Posledným povinným parametrom je „*protocol*“. Tento parameter definuje transportný protokol nesený v IP, a môže nadobúdať hodnoty TCP, UDP a ICMP.

Z nepovinných parametrov môže administrátor definovať „*description*“, čo je popis definovaných pravidiel a dvojica „*sourceAddressPrefix*“ a „*sourcePortPrefix*“. Tie definujú zdrojovú IP adresu a zdrojový port. Zdrojová adresa a port sú dostačujúce, pretože ak by administrátor potreboval definovať cieľovú adresu a port, môže jednoducho zmeniť definovaný smer pravidiel.

Obdobne ako v systémoch AWS a OpenStack, ani systém Azure nedovoľuje zakazovať pravidlami prevádzku. Všetky pravidlá prevádzku povoľujú, a na konci pravidiel je vždy implicitný zákaz všetkého, čo nebolo pravidlami povolené. Na obrázku Obrázok 27 sú znázornené parametre bezpečnostnej skupiny v systéme Azure.



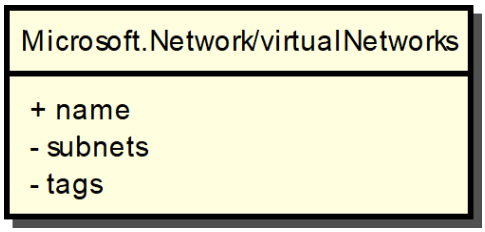
Obrázok 27 Bezpečnostná skupina v systéme MS Azure, zdroj autor

Sieť

Systém Azure má odlišnú filozofiu prideľovania podsietí do sietí, ako systémy AWS a OpenStack. V AWS a OpenStack môže administrátor prideliť podsieť do siete. To znamená, že v definícii podsiete je špecifikované, do akej siete bude patriť. V systéme Azure je to opačne, v sieti sa nachádza definícia, ktoré podsiete do nej patria.

Názov definície siete v systéme Azure je „*Microsoft.Network/virtualNetworks*“. Sieť má jeden povinný parameter – meno siete nazvaný „*name*“. Okrem mena sa tu nachádzajú dva nepovinné parametre. Prvý je „*subnets*“, ktorý definuje zoznam podsietí patriacich do siete. Druhý parameter sú značky, ktoré môže administrátor prideliť ku sieti a nazýva sa „*tags*“.

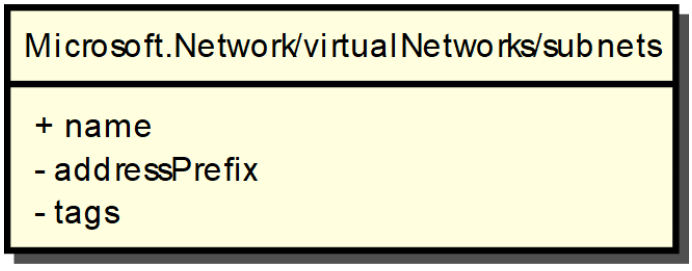
Rovnako ako v systéme OpenStack, ani v systéme Azure nie je nutné, aby podsiete jednej siete patrili do definovaného rozsahu, ktorý je uvedený v definícii siete. Diagram závislostí entity sieť v systéme Azure sa nachádza na obrázku Obrázok 28.



Obrázok 28 Sieť v systéme MS Azure, zdroj autor

Podsieť

Podsieť sa v systéme Azure nachádza v rovnakej časti, „*virtualNetworks*“ a definuje ako „*Microsoft.Network/virtualNetworks/subnets*“. Má tri parametre, z toho je povinné len meno podsiete, nazvané „*name*“. Nepovinný parameter „addressPrefix“ definuje adresový rozsah podsiete vo formáte CIDR. Posledným parametrom je „*tags*“, ktorý obdobne ako pri všetkých ostatných entitách dovoľuje administrátorom pridať ku konkrétnej podsieti značky, ktoré môže administrátor využité pre jednoduchšie identifikovanie entity. Parametre podsiete v systéme Azure sa nachádzajú na obrázku Obrázok 29.



Obrázok 29 Podsieť v systéme MS Azure, zdroj autor

4.3 Návrh všeobecného modelu

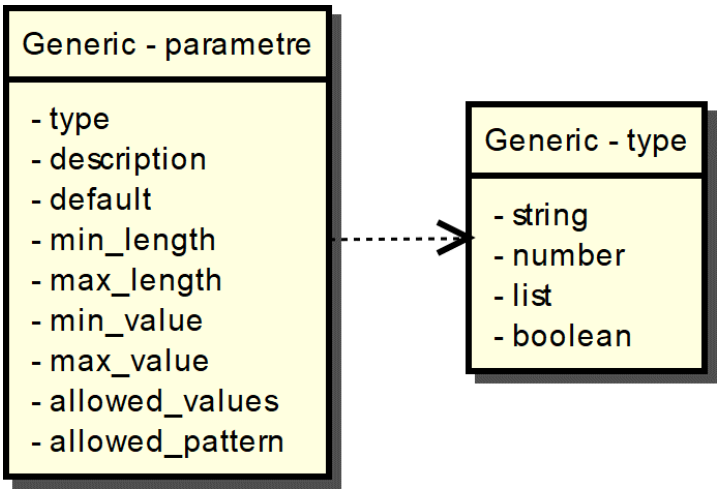
Na základe analýzy existujúcich IaaS CC riešení je ďalej spracovaný návrh všeobecného modelu, ktorý je záverečnou a podstatnou časťou riešenia práce. V terminológii MDA tento model korešponduje s vrstvou PIM a je vytváraný na základe analýzy vrstiev PSM vybraných CC riešení. Navrhovaný všeobecný model obsahuje entity IaaS služby, ich parametre a vzťahy, ktoré sú nezávislé od konkrétnej implementácie IaaS služby. Model bude flexibilne rozšíriteľný a implementovaný ako nástroj použiteľný pre riešenie portability IaaS služieb.

Pri návrhu sme vychádzali z vyššie popísaných existujúcich systémov a z dôvodu jednoduchšej tvorby transformačných pravidiel. Návrh logických entít pre všeobecný model IaaS služby je spracovaný v nasledujúcich podkapitolách.

4.3.1 Parametre prostredia

V navrhovanom všeobecnom modeli je riešená podpora pre parametre skriptov. Obdobne ako v systémoch AWS a OpenStack, aj v tomto modeli môžu mať logické entity parametrov vlastné parametre. Jedným z nich je typ, ktorý označuje vlastné hodnoty parametra. Ďalej je potrebný popis entity parametra a implicitná hodnota, ktorá bude použitá v prípade, ak administrátor nedefinuje novú hodnotu pri aplikovaní skriptu. Ako aj v konkrétnych systémoch, aj vo všeobecnom modeli sú obmedzujúce faktory pre vlastnú hodnotu parametra. Môže to byť dĺžka reťazca, interval hodnôt pre číselné hodnoty, regulárny výraz obmedzujúci znaky v reťazci, alebo zoznam predvolených hodnôt.

Na obrázku Obrázok 30 sa nachádza logické členenie parametra v navrhovanom všeobecnom modeli. Entita „*Generic - type*“ zobrazuje hodnoty, ktoré môžu byť v parametri *type*. Toto vyčlenenie je urobené z hľadiska názornosti a pochopenia konceptu parametra.

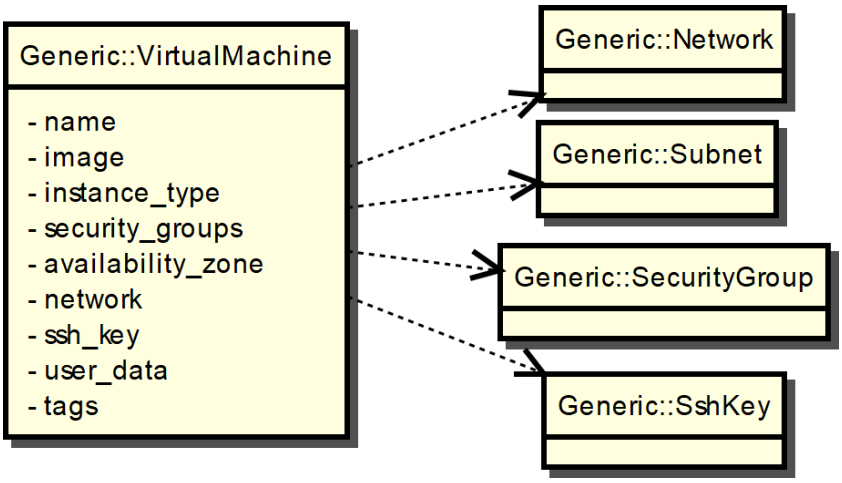


Obrázok 30 Parametre vo všeobecnom systéme, zdroj autor

4.3.2 Virtuálny stroj

Medzi CC systémami AWS a OpenStack je minimálny rozdiel. OpenStack má navyše názov VM, čo je možné do systému AWS jednoducho pretransformovať pomocou parametra *Tags*. Odlišnosť je v tom, že v systéme AWS nie je možné priradiť VM do celej siete, a teda ani do všetkých jej podsietí. VM môže byť priradená len do konkrétnych podsietí. Ak je v systéme OpenStack VM pripojená do celej siete, vieme toto pripojenie interpretovať ako pripojenie do všetkých podsietí danej siete a následne jednoducho transformovať do systému AWS.

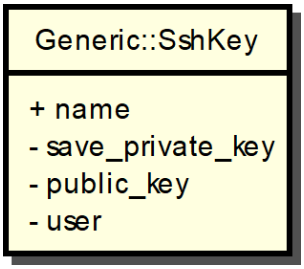
Diagram závislostí virtuálneho stroja navrhovaného všeobecného modelu na obrázku Obrázok 31 je identický s diagramom systému OpenStack. Keďže sa v ňom nenachádzajú žiadne povinné parametre, na rozdiel od systému OpenStack sa v ňom nenachádza logická entita šablóny výpočtových zdrojov pre VM. Rozdiely v diagramoch medzi CC systémami AWS a OpenStack je možné pomerne jednoducho vyriešiť. Z toho dôvodu si môžeme dovoliť prevziať popis jedného systému, ktorý má v niektorých ohľadoch väčšie možnosti konfigurácie.



Obrázok 31 VM vo všeobecnom modeli, zdroj autor

4.3.3 SSH kľúč

Pri návrhu všeobecného modelu sme vychádzali zo systémom OpenStack. Dôvodom je tá skutočnosť, že funkcionalita v systéme OpenStack je plne postačujúca pre akékoľvek IaaS prostredie. Na obrázku Obrázok 32 sa nachádza diagram závislostí SSH kľúča v navrhovanom všeobecnom modeli. Rovnako ako v systéme OpenStack je jediným povinným parametrom meno kľúča.

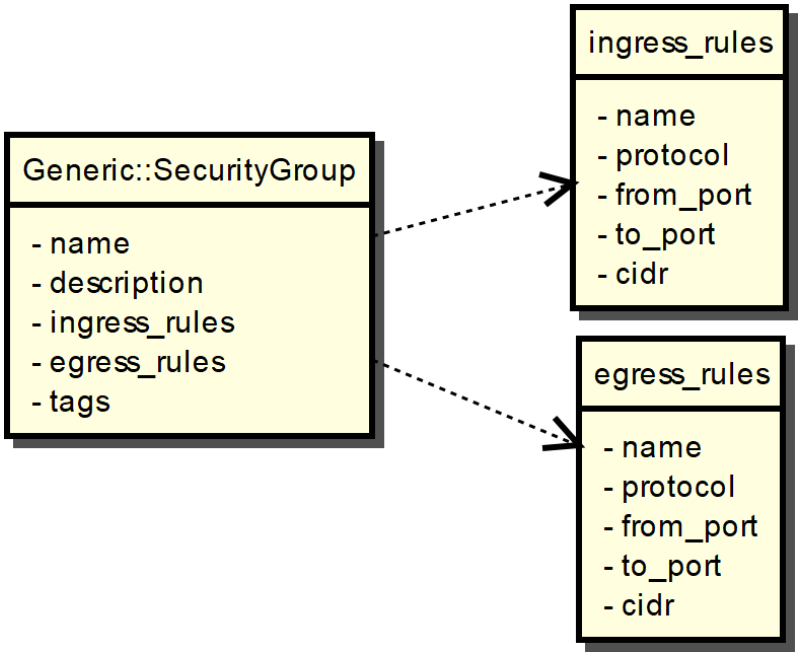


Obrázok 32 SSH kľúč vo všeobecnom modeli, zdroj autor

4.3.4 Bezpečnostná skupina

Pre návrh všeobecného modelu sme využili systém AWS. Vyčlenenie bezpečnostných pravidiel do samostatných logických entít AWS dovoľuje väčšiu variabilitu konfigurácií ako priame integrovanie pravidiel do bezpečnostnej skupiny. Z toho dôvodu navrhujeme vyčleniť skupiny bezpečnostných pravidiel ako samostatné logické entity, čo je

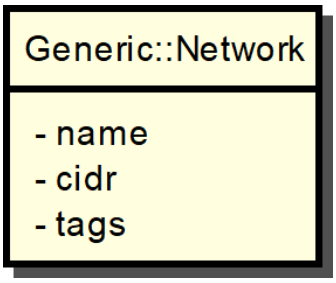
zobrazené aj na obrázku Obrázok 33. Bezpečnostná skupina má podobne ako v systémoch AWS a OpenStack parametre meno, popis, tagy a vlastné bezpečnostné pravidlá. Tie sú rozdelené do dvoch entít, samostatne pre prichádzajúce a samostatne pre odchádzajúce pakety. Skupiny pravidiel majú totožnú syntax, nachádza sa v nich meno, typ transportného protokolu neseného v pakete, dve hranice intervalu transportných portov a IP prefix v CIDR formáte. Diagram závislostí bezpečnostnej skupiny sa nachádza na obrázku Obrázok 33, kde môžeme vidieť že skupina nemá žiadne externé závislosti a môže existovať ako úplne samostatná entita.



Obrázok 33 Bezpečnostná skupina vo všeobecnom modeli, zdroj autor

4.3.5 Sieť

Vo všeobecnom modeli sme zjednotili atribúty zo všetkých systémov tak, aby boli pokryté najčastejšie používané parametre v každom zo systémov V entite sieť je jediný povinný parameter CIDR v systéme AWS. Na obrázku Obrázok 34 sa nachádza model závislostí siete vo všeobecnom modeli.

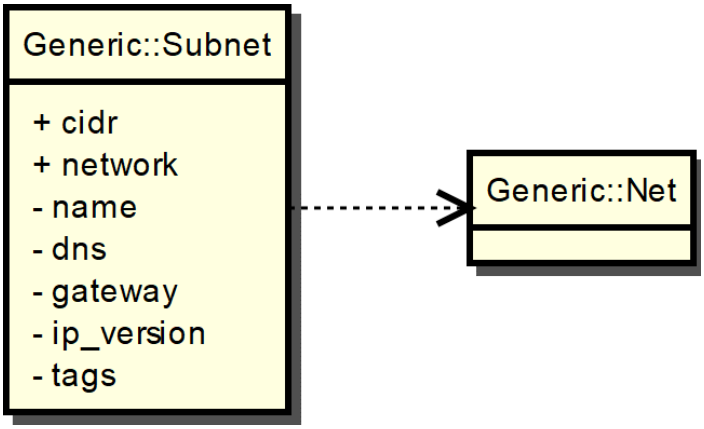


Obrázok 34 Sieť vo všeobecnom modeli, zdroj autor

4.3.6 Podsieť

Vo všeobecnom modeli sme sa navrhli entity tak, aby bola možná transformácia z/do ľubovoľného CC systému. Pretože systém AWS je podmnožinou systému OpenStack, zvolili sme parametre v zhodné so systémom OpenStack.

Na obrázku Obrázok 35 je znázornený diagram závislostí podsiete vo všeobecnom modeli. Znovu je tu jediná entita, na ktorej podsieť závisí – entita siete. Tá opäť musí existovať , aby mohla združovať podsiete.



Obrázok 35 Podsieť vo všeobecnom modeli, zdroj autor

4.4 Transformačné pravidlá

Táto časť práce tvorí návrh transformačných pravidiel riadenia transformácii medzi všeobecným systémom a jednotlivými implementáciami CC systémov. Transformácie sú vytvorené za účelom demonštrácie riešenia len pre základné entity IaaS služby demonštrujúce ideu návrhu. Zvyšné entity je možné v budúcnosti doplniť, ich popisom

a návrhom odpovedajúcich transformácií. Návrh transformácií je flexibilný a rozšíriteľný podľa budúcich vznikajúcich potrieb či pre pridávanie ďalších IaaS prostredí.

4.4.1 Parametre prostredia

V tabuľke Tabuľka 2 sa nachádzajú transformácie pre parametre prostredia. Z tabuľky vyplýva, že systém AWS môžeme priamočiaro transformovať do všeobecného modelu. Systém OpenStack je zložitejší, pretože obmedzenia ako dĺžka reťazca či interval číselných hodnôt sa nachádzajú zapuzdrené v časti *constraints*. Systém Azure je rovnako ako systém AWS priamo transformovateľný do všeobecného modelu. Výnimku tvorí parameter *description*, ktorý je o jednu úroveň nižšie v časti *metadata*. Parameter *allowed_pattern*, ktorý definuje regulárnym výrazom použiteľné hodnoty, nie je v systéme Azure implementovaný.

Parametre *length* a *range* v časti *constraints* obsahujú dve hodnoty – *min* a *max*. Tieto hodnoty tak vieme priamo transformovať do hodnôt *min_length* a *max_length*, resp. *min_value* a *max_value*. Hodnoty *allowed_values* a *allowed_pattern* sú transformovateľné priamo napriek tomu, že v CC systéme OpenStack sú o jednu úroveň nižšie – v rámci parametra *constraints*.

Tabuľka 2 Transformačné pravidlá pre parametre prostredia, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
type	Type	type	type
description	Description	description	metadata - description
default	Default	default	defaultValue
min_length	MinLength	constraints - length	minLength
max_length	MaxLength	constraints - length	maxLength
min_value	MinValue	constraints - range	minValue
max_value	MaxValue	constraints - range	maxValue
allowed_values	AllowedValues	constraints - allowed_values	allowedValues
allowed_pattern	AllowedPattern	constraints - allowed_pattern	-

V tabuľke Tabuľka 3 sú uvedené transformácie špeciálne pre parameter *type*. Tento parameter je dôležitý, pretože na jeho základe sa môžu niektoré hodnoty ignorovať. Ak bude tento parameter obsahovať hodnotu *number*, hodnoty obmedzujúce dĺžku reťazca a regulárny výraz obmedzujúci znaky reťazca budú ignorované ako v systéme AWS, tak v systéme OpenStack.

Môžeme konštatovať, že všetky hodnoty sú priamo transformovateľné. Jedinú výnimku tvorí hodnota *boolean*, ktorú systém AWS vôbec nepodporuje. Pri tejto hodnote nie je možné plnohodnotne automatizovať transformáciu, a je potrebné informovať administrátora, aby vykonal manuálnu kontrolu a prerobenie popisného skriptu.

Tabuľka 3 Transformačné pravidlá pre parametre prostredia - typ, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
string	String	string	string
number	Number	number	int
list	List	comma_delimited_list	array
boolean	-	boolean	bool

4.4.2 Virtuálny stroj

Tabuľka 4 znázorňuje transformačné pravidlá virtuálnych strojov medzi CC systémami OpenStack, AWS, Azure a všeobecným modelom. CC systém AWS neobsahuje názov VM, meno VM je možné nastaviť v parametri *Tags*.

Ďalší rozdiel je v šiestom parametri, kde všeobecný model a systém OpenStack majú uvedenú sieť, systém AWS má podsieť a v systéme Azure je možné definovať priamo rozhranie, ktoré je možné následne pripojiť do konkrétnej siete, alebo podsiete. Ak v systéme OpenStack pripojíme VM do celej siete, znamená to pripojenie do všetkých jej podsietí. Taktiež je možné pripojiť VM len do jednej konkrétnej podsiete. Môžeme to označiť ako priamu transformáciu, kde pripojenie do celej siete zameníme za niekoľkonásobné pripojenie do všetkých podsietí.

Ako môžeme vidieť v tabuľke Tabuľka 4, CC systém Azure má oproti systémom OpenStack a AWS značne zložitejšiu štruktúru. Napriek tomu, sú tieto parametre

významovo identické, takže transformácia do všeobecného modelu je priama, je len potrebné dbať iba na správne umiestnenie parametrov v štruktúre skriptu.

Tabuľka 4 Transformačné pravidlá pre virtuálny stroj, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
name	-	name	name
properties - image	properties - ImageId	properties - image	properties - StorageProfile - imageReference - id
properties - instance_type	properties - InstanceType	properties - flavor	properties - hardwareProfile - vmSize
properties - security_groups	properties - SecurityGroupIds	properties - security_groups	-
properties - availability_zone	properties - AvailabilityZone	properties - availability_zone	zones
properties - network	properties - SubnetId	properties - networks	properties - networkProfile - networkInterfaces - id
properties - ssh_key	properties - KeyName	properties - key_name	properties - OSProfile - linuxConfiguration - ssh - publicKeys - keyData
properties - user_data	properties - UserData	properties - user_data	properties - OSProfile - CustomData
properties - tags	properties - Tags	properties - tags	tags

4.4.3 SSH kľúč

V systémoch AWS a Azure, ako je uvedené časti 4.2.1, nie sú špecifikované entity SSH kľúč, preto úplne absentujú akékoľvek pravidlá. Kľúč nie je možné vytvoriť v rámci skriptu, ale len cez webové rozhranie, či priamy import hodnoty kľúča. Keďže všeobecný model bol inšpirovaný systémom OpenStack, v zásade nie je nutné robiť žiadne transformácie medzi týmito modelmi. Pretože kľúčové slová sú rovnaké, stačí jednoducho skopírovať hodnoty.

Tabuľka 5 Transformačné pravidlá pre SSH kľúč, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
properties - name	-	properties - name	-
properties - save_private_key	-	properties - save_private_key	-
properties - public_key	-	properties - public_key	-
properties - user	-	properties - user	-

4.4.4 Bezpečnostná skupina

Systém OpenStack má pravidlá integrované do jednej logickej entity, zatiaľ čo AWS a Azure tieto pravidlá vyčlenili do samostatných logických entít. V tabuľke Tabuľka 6 je uvedené, že pravidlá nemajú spoločný prienik vo všeobecnom modeli. Systém OpenStack používa jeden parameter *rules*, zatiaľ čo všeobecný model, systém AWS a Azure používajú oddelené parametre pre vstupné a výstupne pravidlá. AWS navyše rozdeľuje tieto entity pre vstupné a výstupné pravidlá, zatiaľ čo Azure používa jednu spoločnú logickú entitu pre obidva smery. Vo všeobecnom modeli sme navrhli dva nezávislé parametre, podobne ako v systéme AWS. Z toho dôvodu je možné priame transformovanie parametrov medzi systémom AWS a všeobecným modelom. Pri transformácii medzi systémom OpenStack a všeobecným modelom bude potrebné vytvárať nové logické entity, respektíve združiť viac entít do jednej. Mapovanie do systému Azure je takmer priame, je potrebné len rozdeliť pravidlá na vstupné a výstupné entity podľa parametra *direction*, ktorý sa nachádza v každej entite pravidiel.

Posledný parameter *tags* zatiaľ nie je v systéme OpenStack implementovaný. Ak by sa vývojári rozhodli implementovať tento parameter v nasledujúcich verziách, bude potrebné doplniť aj túto transformáciu, čo ale nepredstavuje výraznejší problém.

Tabuľka 6 Transformačné pravidlá pre bezpečnostnú skupinu, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
properties - name	properties - GroupName	properties - name	name
properties - description	properties - GroupDescription	properties - description	-
-	-	properties - rules	securityRules

properties - ingress_rules	properties - SecurityGroupIngress	-	-
properties - egress_rules	properties - SecurityGroupEgress	-	-
properties - tags	properties - Tags	-	tags

Nasledujúca Tabuľka 7 je združenou tabuľkou. Pre systém AWS a pre bezpečnostnú skupinu sú v nej uvedené transformácie logických entít pravidiel pre prichádzajúce a odchádzajúce pakety, zatiaľ čo pre systém OpenStack to sú pravidlá integrované v tele bezpečnostnej skupiny. Ako vyplýva z tabuľky Tabuľka 7, transformácie týchto parametrov sú takmer priame. Rozdiel je v parametri mena skupiny, čo systém OpenStack nepotrebuje, pretože sa nachádza priamo v tele bezpečnostnej skupiny. Systém OpenStack má však parameter smeru v ktorom sa budú pakety porovnávať s pravidlom. Ten ale nie je potrebný vo všeobecnom modeli a v systéme AWS, pretože tieto skupiny sa priamo nasadzujú v požadovanom smere toku.

Tabuľka 7 Transformačné pravidlá pre bezpečnostnú skupinu - pravidlá, zdroj autor

Všeobecný model	**AWS**	**OpenStack**	**Azure**
name	GroupName	-	name
description	-	-	description
-	-	direction	direction
protocol	IpProtocol	protocol	protocol
from_port	FromPort	port_range_min	sourcePortPrefix
to_port	ToPort	port_range_max	-
cidr	CidrIp	remote_ip_prefix	sourceAddressPrefix

4.4.5 Sieť

V tabuľke Tabuľka 8 sa nachádzajú transformácie jednotlivých atribútov siete do všeobecného modelu. Ak systém nepoužíva niektorý z parametrov, je parameter označený pomlčkou.

Tabuľka 8 Transformačné pravidlá pre sieť, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
properties - name	properties - VpcId	properties - name	name
properties - subnets	-	-	subnets
properties - cidr	properties - CidrBlock	-	-
properties - tags	properties - Tags	properties - tags	tags

4.4.6 Podsieť

Z transformačných pravidiel podsiete v časti 4.2 vyplýva, že v systémoch AWS a Azure je podsieť konfigurovateľná menším počtom parametrov ako v systéme OpenStack. Čiastočne to ale vyplýva z povahy systémov AWS a Azure, kde poskytovateľ vopred určil niektoré hodnoty a používateľ ich nemôže meniť, ako napríklad predvolená brána má v oboch systémoch vždy prvú možnú adresu z podsiete. Taktiež nie je možné určiť verziu IP protokolu, pretože v súčasnosti AWS aj Azure ponúkajú len IPv4 konektivitu.

Tabuľka 9 Transformačné pravidlá pre podsieť, zdroj autor

Všeobecný model	AWS	OpenStack	Azure
properties - name	-	properties - name	name
properties - network	properties - VpcId	properties - network	-
properties - cidr	properties - CidrBlock	properties - cidr	addressPrefix
properties - dns	-	properties - dns_nameservers	-
properties - gateway	-	properties - gateway_ip	-
properties - ip_version	-	properties - ip_version	-
properties - tags	properties - Tags	properties - tags	tags

4.5 Implementácia a overenie

V nasledujúcej časti je uvedená praktická implementáciu navrhnutého všeobecného modelu. Do úvahy prichádzalo niekoľko možností implementácie. Z rôznych programovacích jazykov sme sa rozhodli pre jazyk Python, konkrétne pre verziu Python3. Je to z toho dôvodu, že tento jazyk je dostatočne vysokoúrovňový, interpretovaný, a tým pádom platformovo nezávislý.

Interpretovaný jazyk nie je potrebné kompilovať pred spustením. Namiesto toho, aby boli zdrojové kódy skompilované do inštrukcií procesora, sú tieto kódy odovzdané tzv. interpreteru. Interpreter je program, ktorý transformuje zdrojové kódy počas behu programu do inštrukcií procesora. Takýmto prístupom môže programátor zmeniť zdrojové kódy bez toho, aby bolo potrebné celý program kompilovať, čím sa môže urýchliť vývoj a ladenie programu. Tým, že je program interpretovaný je ho možné spustiť na akomkoľvek zariadení, ktoré má k dispozícii interpreter. Pri tomto prístupe treba zároveň dbať o to, že aplikácie náročné na výkon môžu mať problémy, pretože ich zdrojové kódy musia zakaždým prejsť transformáciou. Réžia interpretácie môže zaberať podstatnú časť výkonu aplikácie.

Platformová nezávislosť znamená, že vytvorený program je možné spustiť na rôznych hardvérových platformách a architektúrach. To znamená že nie je potrebné upravovať alebo kompilovať program pre každú architektúru zvlášť, stačí program preniesť a spustiť na cieľovom systéme.

Dodržali sme tie konvencie, ktoré sa používajú pri písaní zdrojových kódov. Dôvodom bola snaha o sprehľadnenie kódu. Názvy tried sa začínajú veľkým, názvy metód malým začiatočným písmenom. Atribúty začínajú písmenom „a“, parametre začínajú písmenami „pa“. Názvy premenných sú s malým začiatočným písmenom a používajú tzv. „ťaviu notáciu“. Tá spočíva v tom, že ak je názov premennej viacslovný, každé slovo začína veľkým písmenom. Ako príklad uvedieme pramennú „pocet novych objektov“, ktorá by sa v danej notácii zapísala ako „pocetNovychObjektov“. Keďže predpokladáme, že aplikácia bude aj naďalej vyvíjaná a vylepšovaná nielen na Slovensku, zdrojové kódy vrátane komentárov sú písané v anglickom jazyku. Kompletné zdrojové kódy sa nachádzajú v prílohe na CD. Taktiež sú voľne dostupné na webovom úložisku BitBucket, čo je verejne dostupný repozitár.

Aplikácia je navrhnutá modulárne. Znamená to, že je vytvorené jadro aplikácie, do ktorého sú následne jednotlivé moduly CC IaaS implementácií pridávané. Hlavnou

triedou je „CloudMigration“, z ktorej sú volané funkcie ostatných tried. Predstavuje rozhranie, cez ktoré je možné ovládať aplikáciu.

Ďalšou triedou je „MainWindow“, ktorá obsahuje grafické rozhranie k aplikácii a pomocou nej používateľ interaguje s aplikáciou. Z tejto triedy sú následne volané funkcie triedy „CloudMigration“. Z tohto prístupu je zrejmé, že táto trieda môže byť nahradená akoukoľvek inou triedou, ktorá by ovládala aplikáciu. V budúcnosti by tak mohlo pribudnúť ovládanie aplikácie napríklad cez príkazový riadok (CLI – Command Line Interface), alebo vytvorením API (Application Program Interface).

Trieda „Loader“ slúži na inicializovanie modulov a transformačných pravidiel, ktoré sa budú používať. Táto trieda taktiež inicializuje triedy, ktoré tvoria moduly a obsluhujú jednotlivé implementácie CC IaaS riešení (Generic, OpenStack, AWS a Azure). Každá implementácia CC IaaS riešenia má vlastnú triedu, ktorá obsluhuje transformácie z/do všeobecného modelu.

Priečinok „Mapper“ obsahuje textové súbory s definíciami priamych mapovaní zdrojov. Ku každému zdroju, ako napríklad podsieti, je vytvorený súbor, kde sú mapovania parametrov v rôznych implementáciách CC systémov. Ak niektorý systém neobsahuje parameter, je nahradený pomlčkou. Pre názornosť uvedieme príklad časť súboru pre podsieť.

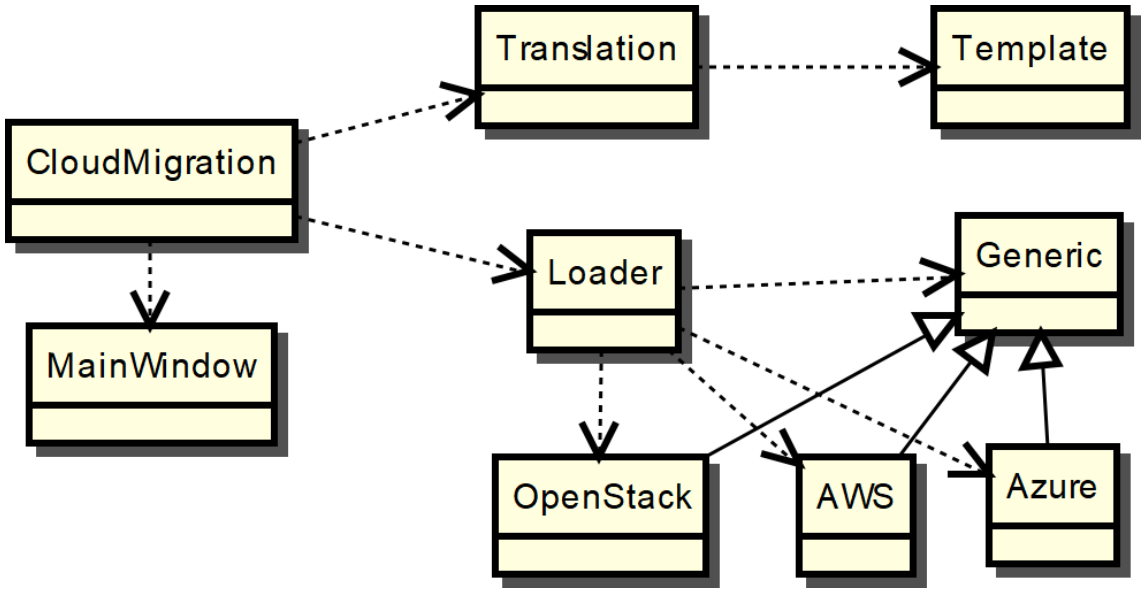
```
Generic     name   network     cidr
OpenStack   name   network     cidr
AWS         -      VpcId       CidrBlock
Azure       name   -           address-prefix
```

Ako môžeme v príklade vidieť, systém AWS nemá definované meno pre podsieť, a systém Azure nedefinuje sieť, do ktorej daná podsieť patrí. Keby sme potrebovali transformovať IP prefix zo všeobecného modelu (v ukážke „cidr“) do systému Azure, názov parametra by sme zmenili za „address-prefix“. Tieto mapovania zodpovedajú mapovacím tabuľkám, uvedeným v časti 4.4.

Trieda „Template“ je zodpovedná za načítanie schém, ktoré sa nachádzajú v priečinku „Schemas“. Každý modul má vlastnú schému parametrov, ktoré sa načítajú a následne podľa týchto schém prebieha preklad. Schémy sú písané v jazyku JSON. V schémach sú definované vlastnosti jednotlivých parametrov, ako ich stručný popis a ich typ. Typ môže momentálne nadobúdať 3 hodnoty – „value“, „list“ a „special“. Pokiaľ je

hodnota „value“, transformácia prebehne priamo, to znamená hodnota sa jednoducho skopíruje. Podobne je to pri hodnote „list“, ktorá definuje, že hodnota je v skutočnosti zoznam hodnôt. Posledným typom je „special“, ktorá hovorí, že nie je možné urobiť priame mapovanie, ale je nutné špeciálne ošetriť mapovanie v kóde. Ako príklad môžeme uviesť špeciálne hodnoty, napríklad verziu skriptu, ktorá je špecifická pre konkrétne CC platformy. V tomto prípade sa vo výslednom skripte zobrazí varovanie, že administrátor by mal manuálne upraviť a skontrolovať parameter.

UML diagram tried sa nachádza na obrázku Obrázok 36.

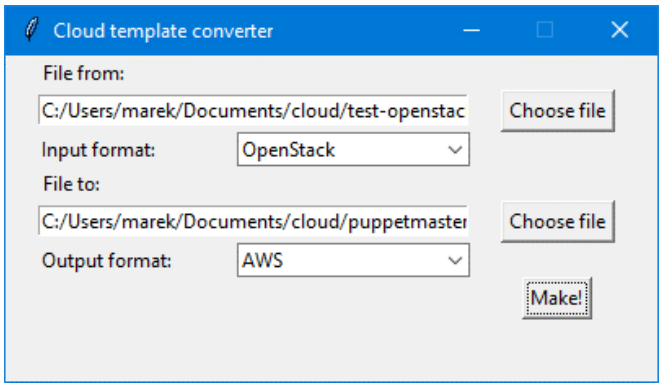


Obrázok 36 UML diagram tried, zdroj autor

Ak v budúcnosti pribudne modul implementujúci transformácie popisu iného CC prostredia do všeobecného modelu, pre úspešné pripojenie k aplikácii potrebuje tento modul spĺňať tri podmienky. Prvou je doplnenie transformácií parametrov do všetkých zdrojov v priečinku „Mapper“. Druhou je vytvorenie schémy konkrétnych parametrov v priečinku „Schemas“ a treťou podmienkou je vytvorenie triedy obsluhujúcej špeciálne prípady transformácií, ktoré nie je možné vykonať priamo.

Na obrázku Obrázok 37 sa nachádza grafické rozhranie aplikácie. Používateľ si môže definovať zdrojový a cieľový súbor, a taktiež CC prostredie, z/do ktorého sa udeje transformácia. Používateľ má na výber všeobecný model, CC systémy OpenStack, Amazon AWS a Microsoft Azure. Tieto systémy si vyberá z rozbaľovacieho menu, ktoré sa nachádza

v okne aplikácie. Keďže dúfame, že naša aplikácia bude osožná administrátorom nielen na Slovensku, grafické rozhranie je vytvorené v anglickom jazyku.



Obrázok 37 Grafické rozhranie programu, zdroj autor

4.5.1 Overenie modelu

Po implementácii je potrebné vykonať verifikáciu modelu a implementácie. V našom prípade sme uskutočnili transformáciu popisov z/do všeobecného modelu. Pre syntaktickú kontrolu transformovaných popisov sme použili reálne implementácie CC systémov, ktoré nám garantujú správnosť transformácie. Ako referenčnú topológiu sme použili jednu virtuálnu inštanciu servera v sieti za logickým smerovačom, ktorá bola znázornená aj na obrázku Obrázok 13.

Vykonali sme transformácie skriptov zo všetkých troch modelov do všeobecného modelu, a následne zo všeobecného modelu spätne do konkrétneho modelu. Ako overenie prikladáme skripty popisujúce našu testovaciu topológiu. Všetky skripty, ktoré popisujeme v nasledujúcej časti sa nachádzajú v prílohe na CD v priečinku „Skripty“ a sú napísané v značkovacom jazyku YAML.

Prvým krokom bolo vytvorenie skriptu s testovacou topológiou. V našom prípade sme ako prvý systém pre transformáciu použili OpenStack, pretože s ním máme najviac skúseností. Skript pre systém OpenStack má názov „OpenStack.yml“.

Po syntaktickom overení skriptu v systéme OpenStack nasledoval druhý krok, transformácia tohto skriptu do všeobecného modelu. Tento pretransformovaný skript sa nachádza v prílohe na CD s názvom „Generic-z-OpenStack.yml“.

Tretím krokom bola transformácia skriptu vo všeobecnom formáte spätne do formátu pre systém OpenStack. Ten sa taktiež nachádza v prílohách s názvom „OpenStack-z-Generic.yml“. Tento súbor sme následne úspešne syntakticky validovali v systéme OpenStack, čím sme potvrdili správnosť a funkčnosť našej implementácie všeobecného modelu.

Rovnakou trojkrokovou metódou sme overili aj syntaktickú správnosť prekladu pre systémy Amazon AWS a Microsoft Azure. V prílohách na CD sa v priečinku „Skripty“ nachádzajú popisné skripty prostredí v systémoch AWS a Azure. Taktiež sa tam nachádzajú tieto skripty pretransformované do všeobecného modelu, a skripty, ktoré sú do týchto implementácií pretransformované spätne zo všeobecného modelu. V obidvoch prípadoch, pri AWS aj Azure, sme použili reálnu implementáciu CC prostredia pre overenie správnosti prekladu daného skriptu. V obidvoch prípadoch sme transformovali rovnakú testovaciu topológiu, ktorá pozostávala z jedného virtuálneho stroja, ktorý sa nachádza vo virtuálnej sieti za logickým smerovačom.

Na základe testov syntaktickej správnosti na reálnych implementáciách CC IaaS systémov konštatujeme, že implementácia všeobecného modelu vytvorená v tejto dizertačnej práci, ako aj transformačných pravidiel je funkčná a pripravená na používanie. Pre reálne použitie je však potrebné budúce rozšírenie všeobecného modelu o ďalšie entity.

4.6 Diskusia výsledkov

V súčasnosti sú dostupné pre nasadenie CC s IaaS rovnocenné riešenia, kde ich konkrétny výber môže byť podmienený rôznymi faktormi (jednoduchosť používania, rozsiahlosť dokumentácie, jednoduchosť nasadenia v prípade privátneho CC, ...), a jeden z nich môže byť aktuálna cena. Tá sa však môže mnohokrát rýchlo vyvíjať a pri dostupnosti alternatívnych riešení s lepšou cenou pri rovnakých ponúkaných službách pripadajú do úvahy aj aspekty prechodu od jedného poskytovateľa CC k inému. Z tohto pohľadu vystupujú do popredia otázky prenositeľnosti svojej IaaS služby od poskytovateľa k poskytovateľovi a potencionálne s tým už uvádzaný problém *vendor lock-in*.

Navrhnutý spôsob transformácií umožní systémovo riešiť problematiku prenosu IaaS služby medzi rôznymi IaaS prostrediami. Pomocou nástroja, ktorý obsahuje navrhovaný všeobecný model, platformovo závislé modely ako aj navrhované transformačné pravidlá, bude možné vykonávať transformácie popisných skriptov IaaS služby (entít, parametrov

a ich vzťahov) medzi rôznymi CC systémami (medzi všeobecným a závislým modelom a naopak). Práca tak rieši identifikovaný problém portability IaaS služieb. Model je navrhnutý ako všeobecný a do budúcnosti otvorený pre rozširovanie. Vypracovanie špecifikácie uľahčí používateľom aspekt migrácie služby a prispeje k väčšej otvorenosti prostredia CC služieb. Vzhľadom na komplexnosť problematiky sa práca zameriava na iba na IaaS službu, vzniká tu však predpoklad, že po úspešnom overení je využiteľnosť riešenia aj pre iné CC služby. Práca je zároveň príspevkom k oblasti štandardizácie portability.

Veríme, že táto práca rieši otázku problému, ktorý je vo svete aktuálny. Toto tvrdenie podporuje aj skutočnosť, že v decembri roku 2017 spoločnosť ISO vydala štandard 19941 [55], ktorý definuje interoperabilitu a portabilitu z pohľadu poskytovateľov a používateľov CC prostredí.

Záver

Hlavným cieľom dizertačnej práce bolo navrhnutie a otestovanie všeobecného modelu pre skripty popisujúce CC prostredia. Prvým krokom bola analýza existujúcich CC prostredí. Z dôvodu utvorenia komplexnejšieho pohľadu sme analyzovali zástupcov privátnych, ale aj verejných CC IaaS prostredí. Analýza spočívala v pochopení daného riešenia v zmysle logickej stavby topológie. Následným krokom bolo zoznámenie sa s konkrétnymi logickými entitami a získanie zručností s ich praktickým používaním v popisných skriptoch.

Návrh všeobecného modelu vychádzal z analyzovaných CC prostredí tak, aby bol čo najjednoduchší na implementáciu a taktiež, aby čo najpriamejšie transformoval parametre daných CC prostredí. Zároveň bolo našou snahou, aby logická stavba a členenie navrhovaného modelu boli čo najviac podobné pôvodným modelom z toho dôvodu, aby bol administrátorom čo najfamiliárnejší.

Poslednou časťou riešenia bola implementácia všeobecného modelu v programovacom jazyku a následné praktické overenie. Súčasťou implementácie bolo testovanie pretransformovaných skriptov v reálnych implementáciách CC prostredí, čím bola zaručená syntaktická bezchybnosť transformácií.

Za prínos dizertačnej práce považujeme vytvorenie všeobecného modelu popisného skriptu CC IaaS prostredia, jeho implementáciu a praktické overenie, ako aj prehľad aktuálneho stavu poznania CC IaaS služby doma a v zahraničí.

Veríme, že navrhnutý model, ako aj jeho implementácia budú používané pri reálnych transformáciách popisných skriptov a uľahčia prácu administrátorom. Model, ako aj zdrojové kódy implementácie v programovacom jazyku Python3 sú verejne dostupné a predpokladáme, že tento projekt bude aj naďalej rozširovaný a zveľaďovaný.

Možnosti ďalšieho smerovania výskumu a vývoja v CC oblasti sú v stanovení pravidiel transformácií, prípadne štandardizácii niektorého z popisných jazykov ako všeobecne platného a uznávaného, ktorý môže byť použitý v akomkoľvek CC prostredí. Podľa nášho názoru môže prísnejšia štandardizácia pomôcť zjednotiť trh v oblasti CC, čím donúti poskytovateľov k ponúkaniu kvalitnejších služieb, čo bude mať pozitívne dôsledky pre prevádzku používateľských aplikácií.

Zoznam použitej literatúry

[1] G. A. A. Santana, CCNA Cloud, Cisco Press, 2016, p. 609.

[2] European Commission, „Cloud Select Industry Group on Service Level Agreements," [Online]. Available: https://ec.europa.eu/digital-single-market/en/cloud-select-industry-group-service-level-agreements. [Cit. December 2017].

[3] European Commission, „Cloud Select Industry Group on Code of Conduct," [Online]. Available: https://ec.europa.eu/digital-single-market/en/cloud-select-industry-group-code-conduct. [Cit. December 2017].

[4] European Telecommunications Standards Institute, „Cloud Standards Coordination," [Online]. Available: http://csc.etsi.org/. [Cit. December 2017].

[5] European Commission, „Expert Group on Cloud Computing Contracts," [Online]. Available: http://ec.europa.eu/justice/contract/cloud-computing/expert-group/index_en.htm. [Cit. December 2017].

[6] ITU-T, „Y.3500 Overview and vocabulary," August 2014. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3500-201408-I. [Cit. Január 2018].

[7] F. Liu a et.al., „NIST Cloud Computing Reference Architecture, SP-500-292," 8 September 2011. [Online]. Available: https://www.nist.gov/publications/nist-cloud-computing-reference-architecture. [Cit. Február 2017].

[8] Roadmap Working Group, „NIST Cloud Computing Standards Roadmap, SP 500-291 version 2," 2013. [Online]. Available: http://www.cloudwatchhub.eu/sites/default/files/NIST_Cloud-Standards-Roadmap_v2.pdf. [Cit. Január 2017].

[9] „What is a Private Cloud?," [Online]. Available: http://www.interoute.com/cloud-article/what-private-cloud. [Cit. Január 2017].

[10] TechTarget, „Cloud Bursting," Apríl 2017. [Online]. Available: http://searchcloudcomputing.techtarget.com/definition/cloud-bursting. [Cit. Január 2018].

[11] „OpenStack Project,“ [Online]. Available: https://www.openstack.org/. [Cit. marec 2017].

[12] Amazon AWS, „What Is AWS GovCloud (US)?,“ [Online]. Available: http://docs.aws.amazon.com/govcloud-us/latest/UserGuide/whatis.html. [Cit. December 2017].

[13] RightScale, „State of the cloud report,“ 2017. [Online]. Available: http://assets.rightscale.com/uploads/pdfs/RightScale-2017-State-of-the-Cloud-Report.pdf. [Cit. Marec 2017].

[14] „What is a Public Cloud?,“ [Online]. Available: http://www.interoute.com/cloud-article/what-public-cloud. [Cit. Január 2017].

[15] „What is a Hybrid Cloud?,“ [Online]. Available: http://www.interoute.com/cloud-article/what-hybrid-cloud. [Cit. Január 2017].

[16] M. Boniface a et.al, „Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds,“ rev. *Fifth International Conference on Internet and Web Applications and Services*, 2010.

[17] L. Youseff, M. Butrico a D. Da Silva, „Toward a unified ontology of cloud computing,“ *Grid Computing Environments (GCE) Workshop,* pp. 1-10, 2008.

[18] A. Benlian a T. Hess, „Opportunities and risks of software-as-a-service: Findings from a survey of IT executives,“ *Decision Support Systems,* zv. 52, %1. vyd.1, pp. 232-246, 2011.

[19] S. Hopko, Problematika Cloud computing a jeho využitia v riešení pre potreby KIS, Žilina, 2015.

[20] R. L. Krutz a R. D. Vines, Cloud Security: A Comprehensive Guide to Secure Cloud Computing, Wiley Publishing, Inc., 2010.

[21] R. Dimpi a R. R. K., „A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing,“ *International Journal of Advanced Research in Computer Science and Software Engineering,* zv. 4, %1. vyd.6, pp. 458-461, Jún 2014.

[22] Roadmap Working Group, „NIST Cloud Computing Standards Roadmap,“ 2013. [Online]. Available: http://www.cloudwatchhub.eu/sites/default/files/NIST_Cloud-Standards-Roadmap_v2.pdf. [Cit. Január 2017].

[23] ITU-T, „Y.3502 Reference architecture,“ August 2014. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3502-201408-I/en. [Cit. Decemer 2017].

[24] Wikioedia, „Interoperability,“ Január 2018. [Online]. Available: https://en.wikipedia.org/wiki/Interoperability. [Cit. Január 2018].

[25] Techpedia, „Interoperability,“ Január 2018. [Online]. Available: https://www.techopedia.com/definition/631/interoperability. [Cit. Január 2018].

[26] D. Petcu, „Portability and interoperability between clouds: challenges and case study,“ rev. *ServiceWave'11 Proceedings of the 4th European conference on Towards a service-based internet*, Heidelberg, 2011.

[27] Cloud Standards Customer Council, Interoperability and Portability for Cloud Computing: A Guide, 2014, p. 31.

[28] A. V. Parameswaran a A. Chaddha, „Cloud Interoperability and Standardization,“ *SETLabs Briefings*, zv. 7, %1. vyd.7, pp. 19-26, 2009.

[29] M. Kostoska a et.al, „Cloud Computing Interoperability Approaches – Possibilities and Challenges,“ rev. *Proceedings of 5th Balkan Conf. in Informatics*, Novi Sad, Srbsko, 2012.

[30] IEEE, „Cloud Profiles Working Group,“ [Online]. Available: http://standards.ieee.org/develop/wg/CPWG-2301.html. [Cit. November 2017].

[31] IEEE, „Standard for Intercloud Interoperability and Federation,“ [Online]. Available: https://standards.ieee.org/develop/project/2302.html. [Cit. November 2017].

[32] OASIS, „TOSCA,“ [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca. [Cit. November 2017].

[33] K. Bhavya, K. Yamini a V. Sreenivas, „Cloud Services Portability for secure migration,“ *International Journal of Computer Trends and Technology (IJCTT)*, zv. 4, %1. vyd.4, pp. 546-549, 2013.

[34] I. Bojanova, „Cloud Interoperability and Portability II,“ 25 júl 2013. [Online]. Available: https://www.computer.org/web/irena-bojanova/content?g=5970564&type=blogpost&urlTitle=cloud-interoperability-and-portability-ii. [Cit. Október 2017].

[35] The Open Group, „Cloud Portability and Interoperability,“ [Online]. Available: http://www.opengroup.org/cloud/cloud_iop/p4.htm. [Cit. Október 2017].

[36] „PC Encyclopedia,“ [Online]. Available: https://www.pcmag.com/encyclopedia/term/62863/cloud-platform. [Cit. Január 2018].

[37] RightScale, „State of the cloud report,“ 2015. [Online]. Available: http://assets.rightscale.com/uploads/pdfs/RightScale-2015-State-of-the-Cloud-Report.pdf. [Cit. Marec 2017].

[38] RightScale, „State of the cloud report,“ 2016. [Online]. Available: http://assets.rightscale.com/uploads/pdfs/RightScale-2016-State-of-the-Cloud-Report.pdf. [Cit. Marec 2017].

[39] OMG, „Model Driven Architecture – A Technical Perspective,“ OMG, 2001.

[40] M. Kardoš, Transformácia CIM do PIM v modelom riadenej architektúre (MDA), Žilina, 2011.

[41] S. M. Drazen Brdjanin, „Model Driven Techniques for Data Model Synthesis,“ *Electronics,* zv. VOL. 17, %1. vyd.NO. 2, pp. 130-136, 2013.

[42] A. G. Kleppe, J. B. Warmer a W. Bast, MDA explained : the model driven architecture : practice and promise, Addison-Wesley, 2003.

[43] OMG, „CommonWarehouseMetamodel (CWM) Specificatio,“ OMG, 2001.

[44] S. Gyapay a D. Varró, „Automated Algorithm Generation for Visual Control Structures,“ December 2000. [Online]. Available: http://static.inf.mit.bme.hu/pub/TR-12-2000.pdf. [Cit. Október 2017].

[45] D. Varró, G. Varró a P. Apndrás, „Designing the automatic transformation of visual languages,“ *Science of Computer Programming,* zv. 44, %1. vyd.2, pp. 205-227, 2002.

[46] W3C, „XSL Transformations (XSLT) Version 3.0,“ W3C, 8 Jún 2017. [Online]. Available: https://www.w3.org/TR/xslt/. [Cit. November 2017].

[47] „Amazon Web Services,“ [Online]. Available: https://aws.amazon.com/about-aws/. [Cit. Marec 2017].

[48] „Browse All OpenStack Projects,“ [Online]. Available: https://www.openstack.org/software/project-navigator. [Cit. marec 2017].

[49] [Online]. Available: https://www.cloudave.com/wp-content/uploads/2015/02/slide1.jpg. [Cit. December 2017].

[50] „Microsoft Azure,“ [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-azure/. [Cit. marec 2017].

[51] „Azure Stack Development Kit,“ [Online]. Available: https://azure.microsoft.com/en-us/blog/microsoft-azure-stack-is-ready-to-order-now/. [Cit. 2017].

[52] M. Corporation, „Microsoft Azure Stack,“ [Online]. Available: https://azure.microsoft.com/en-us/blog/microsoft-azure-stack-is-ready-to-order-now/. [Cit. 2017].

[53] VMware, Inc., „vSphere and vSphere with Operations Management,“ [Online]. Available: http://www.vmware.com/products/vsphere.html. [Cit. marec 2017].

[54] VMware, Inc., „vRealize Automation,“ [Online]. Available: http://www.vmware.com/products/vrealize-automation.html. [Cit. marec 2017].

[55] ISO, „ISO/IEC 19941,“ December 2017. [Online]. Available: https://www.iso.org/standard/66639.html. [Cit. Január 2018].

[56] M. A. Bokhari, Q. M. Shallal a Y. K. Tamandani, „Cloud Computing Service Models: A Comparative Study,“ *3rd International Conference on Computing for Sustainable Global Development (INDIACom),* pp. 890-895, 16-18 Marec 2016.

[57] Z. Bizoňová, Model Driven E-learning Platform Integration, Žilina, 2008, p. 216.

[58] T. Mens a P. Van Gorp, „A Taxonomy of Model Transformation,“ *Electronic Notes in Theoretical Computer Science,* zv. 152, pp. 125-142, 2006.

[59] „Interoperability and Portability for Cloud Computing: A Guide,“ November 2014. [Online]. Available: http://www.cloud-council.org/deliverables/CSCC-Interoperability-and-Portability-for-Cloud-Computing-A-Guide.pdf. [Cit. Január 2017].

[60] L. Mohan a e. al., „A Comparative Study on SaaS, PaaS and IaaS Cloud Delivery Models in Cloud Computing,“ *International Journal on Emerging Technologies*, pp. 158-160, 2017.

Prílohy

Príloha A: Obsah CD

Priložené CD obsahuje:

- Práca v elektronickej podobe (formát PDF)
- Zdrojové kódy aplikácie (Priečinok „Zdrojové kódy“)
- Ukážky transformovaných skriptov (Priečinok „Skripty“)