Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Borche Paspalovski

Orkestracija storitev v oblaku z VMware vCenter Orchestrator

diplomsko delo
na univerzitetnem študiju

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01864/2012
Datum: 04.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BORCHE PASPALOVSKI**

Naslov: **ORKESTRACIJA STORITEV V OBLAKU Z VMWARE VCENTER ORCHESTRATOR**

CLOUD SERVICE ORCHESTRATION USING VMWARE VCENTER ORCHESTRATOR

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite storitvene in postavitvene modele računalništva v oblaku. Raziščite procese avtomatizacije in orkestracije v oblaku ter opišite orodja. Podrobno predstavite in primerjajte orodja za orkestracijo. Proučite rešitev za orkestracijo VMware vCenter Orchestrator. Prikažite postopek postavljanja storitev v oblaku na VMware platformi. Preverite zmožnosti orkestracije storitev v hibridnem oblaku ter implementirajte osnovne operacije v hibridnem oblaku: dinamična postavitev nove obremenitve, migracija storitev iz zasebnega oblaka v javni oblak ter optimizacija virov v hibridnem oblaku.

Mentor:

prof. dr. Matjaž B. Jurič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Borche Paspalovski, z vpisno številko **63060430**, sem avtor diplomskega dela z naslovom:

Orkestracija storitev v oblaku z VMware vCenter Orchestrator

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Matjaža Branka Juriča
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 3. marca 2013 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Matjažu Branku Juriču za vso pomoč in napotke pri izdelavi diplomskega dela. Posebej se zahvaljujem moji družini, za moralno in finančno pomoč v času študija.

Kazalo

Povzetek

Računalništvo v oblaku je stil zagotavljanja računalniških storitev, ki v današnjem času postaja vse bolj aktualen. Ena izmed pomanjkljivosti platform računalništva v oblaku je nezadostna stopnja avtomatizacije znotraj oblaka kar se lahko reši s pomočjo orkestracije. Orkestracija v oblaku je način, kako definirati naloge za avtomatizacijo procesov v oblaku. Glavni cilj diplomskega dela je bila izdelava orkestracijskih delovnih tokov za postavitev platformske storitve v oblaku z uporabo orodja za orkestracijo VMware vCenter Orchestrator. V uvodu diplomskega dela je opisana motivacija za uporabo orkestracije v računalništvu v oblaku. Za tem sledi opis osnovnih storitvenih in postavitvenih modelov ter opis koncepta avtomatizacije in orkestracije skupaj z njihovim namenom uporabe znotraj oblaka. V sklopu tega poglavja smo naredili tudi pregled obstoječih tehnologij za avtomatizacijo in orkestracijo, opisali njihove ključne funkcionalnosti in primerjali tehnologije za orkestracijo. Nadaljevali smo s podrobnim opisom orodja VMware vCenter Orchestrator, nato pa smo se osredotočili na proces orkestracije storitev v zasebnem oblaku, tako da smo prikazali proces postavitve platformskih storitev. V zadnjem poglavju smo razširili že obstoječe delovne tokove komponente za orkestracijo z namenom postavitve obremenitve oziroma storitve v hibridni oblak. Na strani javne postavitve hibridnega oblaka smo implementirali in izpostavili osnovne funkcije potrebne za orkestracijo.

Ključne besede: Računalništvo v oblaku, avtomatizacija, orkestracija, Storitve v oblaku, VMware vCenter Orchestrator, delovni tok

Abstract

Cloud computing is a computing style, which actuality has increased lately. One of the downsides of cloud computing platforms is the insufficient level of cloud automation. Cloud orchestration is the way of defining tasks, which can automate processes inside the cloud. The main goal of the thesis is the creation of orchestration workflows, used for the deployment of platform services in the cloud with the use of VMware vCenter Orchestrator component. In the introduction we described the motivation of using orchestration in the cloud. This is followed by the description of cloud computing service models and deployment models. The next chapter presents the basic concepts of automation and orchestration in the cloud. Within this chapter we also made an overview of the existing technologies for automation and orchestration, we described their key functionalities and we showed orchestration technologies comparison. Next we gave a detailed overview on the VMware vCenter Orchestrator component. In the following chapter we focused on the process of orchestrating services in the private cloud. In the final chapter we extended the current workflows of orchestration component for the purpose of deploying services into the hybrid cloud. Additionally, we implemented and exposed the basic functionalities for the purpose of hybrid cloud orchestration.

Keywords: Cloud computing, automation, orchestration, Cloud Services, VMware vCenter Orchestrator, workflow

Poglavje 1

Uvod

Računalništvo v oblaku je model, ki spreminja način poslovanja IT podjetji in je v zadnjem času vse bolj aktualen. Ponudniki ciljni stranki omogočajo najem različnih tipov storitev v odvisnosti od virov, ki jih ponujajo. Ne glede na tip storitev se ponudniki soočajo z veliko problemi zaradi zagotavljanja potrebne storitve organizacijam. To je predvsem zaradi nezadostne stopnje avtomatizacije znotraj oblaka v primeru postavljanja storitev. Platforme, ki jih ponudniki uporabljajo za gradnjo zasebnega oblaka, najpogosteje zajemajo določeno stopnjo avtomatizacije infrastrukturne storitve. Izvedba bolj kompleksnih infrastrukturnih storitev, kot je na primer celotna postavitev in konfiguracija virtualnega stroja (angl. Virtual Machine), pa od ponudnika zahteva še veliko dodatnega dela. Če gremo eno nivo višje, torej na platformske storitve, pa je ponujena stopnja avtomatizacije še manjša, saj je potrebno najprej pripraviti ustrezno infrastrukturno storitev in nato to storitev dodatno razširiti s postavljanjem določene programske opreme.

Problem nezadostne stopnje avtomatizacije rešujemo z uporabo orodij za orkestracijo. Ta orodja omogočajo uporabo večine operacij, ki so del platforme oblaka v enotnem procesu in na ta način izvedbo t.i. orkestracije (angl. Orchestration).

Po drugi strani pa je rešitev hibridnega oblaka še vedno v razvoju. To pomeni, da manjše število platform ponuja poleg zasebnega oblaka tudi kom-

ponente za hibridno postavitev. Poleg tega pa je podpora za izvedbo orkestracije storitev v hibridnem oblaku zelo omejena. Orodja za orkestracijo le redko ponujajo gradnike s katerimi lahko izvedemo osnovne operacije v hibridnem oblaku, kot so na primer migracija storitev in optimizacija obremenitev. Problem postavljanja storitev v hibridnem oblaku rešujemo z dodatno implementacijo operacij in dodatno integracijo orodja za orkestracijo z zunanjimi komponentami.

Cilj diplomskega dela je (1) opisati osnovne koncepte orkestracije in avtomatizacije v oblaku, ter poiskati skupne točke in razlike teh dveh področij, (2) izvesti primerjalno analizo obstoječih rešitev za orkestracijo in ugotoviti ključne lastnosti orkestracijskih orodij, (3) opisati osnovno arhitekturo orodja za orkestracijo vCenter Orchestrator, ter (4) izdelati dva praktična primera, ki obravnavata tako zasebni kot hibridni oblak. Pri prvem praktičnem primeru bomo razvili proces postavljanja storitev v zasebnem oblaku in prikazali celotni proces orkestracije, ki bo pripravil in postavil storitev (Apache Tomcat strežnik) v zasebni oblak. Drugi primer pa bo predstavljal izvedbo operacije nad storitvami v hibridnem oblaku. Tako bomo prikazali kako naj ponudnik na avtomatiziran način kreira storitev in s tem odstrani nepotrebne, ponavljajoče se naloge.

Poglavje 2

Računalništvo v oblaku

2.1 Storitveni modeli oblaka

Storitve v oblaku predstavljajo vse tiste vire, ki jih ponudnik organizira kot poslovne rešitve in ponuja uporabnikom v najem. Izraz "Storitve v oblaku" (angl. Cloud Services) je definiran kot koncept uporabljanja ponovno uporabljivih, drobno-zrnatih komponent na ponudnikovem omrežju [1]. Storitve imajo osnovne značilnosti računalništva v oblaku kot so: visoka skalabilnost, enostavna dostopnost, deljenje med več najemnikov, itd. Koncept ponujanja storitve v oblaku lahko razdelimo na dva vidika: vidik ponudnika in vidik uporabnika.

Ponudniki storitev na področju računalništva v oblaku vidijo oziroma iščejo poslovno priložnost pri ponujanju "Nekaj kot storitev" (angl. Anything as a Service - XaaS). Storitev je lahko del njihove infrastrukture, njihovih platform za postavitev aplikacij ali pa dejansko razvitih aplikacij. Ponudniki imajo storitve organizirane v produktnem katalogu storitev, iz katerega si uporabniki izberejo storitev. Kot primer lahko vzamemo Amazon, ki kot eden izmed največjih ponudnikov storitev v oblaku (pionir na tem področju) ponuja računanje - Elastic Compute Cloud [2], shrambo - Simple Storage Service [3], relacijsko podatkovno bazo - Relational Database Service [4], pretakanje vsebine z lokacijskim predpomnjenjem - CloudFront [5], itd.

Za uporabnika storitve predstavljajo zmožnost pridobivanja virov, brez potrebe po upravljanju le-teh. Storitve so postavljene na oddaljenem podatkovnem centru in so dostopne preko internetne povezave. Dostopnost virov oziroma storitev ter pričakovani nivo zmogljivosti sistema je zagotovljen s parametri sporazuma o ravni storitve (angl. Service Level Agreement - SLA), ki so stvar dogovora med ponudnikom in uporabnikom pri najemu storitve. Varnost podatkov in visoka razpoložljivost storitve sta zagotovljeni.

Storitve lahko ločimo glede na nivo abstrakcije oblaka na katerega se storitev nanaša. Na najvišjem nivoju storitve združimo v storitvene modele: infrastruktura kot storitev (angl. Infrastructure as a Service - IaaS), platforma kot storitev (angl. Platform as a Service - PaaS) ter storitev programske opreme (angl. Software as a Service - SaaS). Poleg tega obstaja še veliko drugih tipov storitev, kot na primer storitev shrambe (angl. Storage as a Service - STaaS), varnost kot storitev (angl. Security as a Service - SECaaS), podatki kot storitev (angl. Data as a Service - DaaS), programski vmesnik kot storitev (angl. API as a Service - APIaaS), itd.

V nadaljevanju bomo opisali najbolj osnovne modele storitev, ki so ključni tudi pri orkestraciji.

2.1.1 Infrastrukturne storitve

Infrastrukturne storitve so tesno povezane z računalniškimi viri. Uporaba teh storitev daje občutek, kot da najamemo fizični stroj, medtem ko je le-ta v večini primerov virtualni. Slednje je mogoče zaradi različnih virtualizacijskih platform, ki izkoriščajo fizične vire in jih abstrahirajo kot virtualne vire. Primeri virtualizacijskih platform so VMware vSphere, Citrix XenServer, Microsoft Hyper-V, KVM, itd. Naslednja možnost, ki jo omogočajo nakateri izmed ponudnikov virtualizacijskih rešitev je virtualizacija fizičnih strežnikov preko orodij Physical-to-Virtual - P2V. Primer takšnega orodja je VMware vCenter Converter [6], ki ponuja avtomatizirano pretvorbo fizičnih strežnikov v virtualne stroje.

Infrastrukturna storitev ne pomeni le najem novega virtualnega stroja.

Uporabnik ima možnost spreminjanja nastavitev le-tega. To pomeni, da lahko sam nastavlja vsaj število procesorjev, velikost pomnilnika, velikost diska ter omrežne opreme. Poleg tega si uporabnik sam izbere tudi operacijski sistem in tip arhitekture. Pomembno je, da lahko uporabnik vse te vire najame brez potrebe po ročnih posegih ponudnika. Uporabniku ni treba skrbeti za vzdrževanje in popravljanje fizične opreme ter za posodabljanje programske opreme in operacijskega sistema.

V infrastrukturne storitve so v večini primerov vgrajene funkcionalnosti platforme za upravljanje oblaka (angl. Cloud Platform). To pomeni, da ponudniku infrastrukturnih storitev, v primeru da uporablja že obstoječe rešitve oblaka, ni potrebno razširjati in nadgrajevati te storitve. Ker te storitve zajemajo najnižji nivo oblaka, so uporabniku na voljo kot najbolj osnovne storitve oblaka. Primer takšnih storitev so: omrežne storitve, storitve "Volume" komponente, ponujanje virtualnega stroja kot storitev, itd.

2.1.2 Platformske storitve

Platformske storitve spadajo v naslednji abstrakcijski nivo - nad infrastrukturnega. Storitve v osnovi vsebujejo virtualni stroj na katerem je nameščen ustrezen operacijski sistem, z možnostjo izbire dodatne programske opreme. Storitev v tem primeru pomeni postavitev strežnika nad operacijskim sistemom (na primer spletni, aplikacijski strežnik, ali strežnik podatkovne baze).

Platformske storitve so namenjene razvijalcem programske opreme, z namenom poenostavitve postopka priprave strežnikov za postavljanje aplikacij. Na ta način razvijalci programske opreme dobijo okolje za razvoj in postavitev programske opreme, ki je nastavljiva glede na potrebe aplikacije. Naloga razvijalcev je razvoj in postavljanje aplikacij v oblaku. Potem ko je aplikacija postavljena in jo razvijalec ponuja kot storitev, lahko govorimo o SaaS aplikaciji ali SaaS storitvi. Poleg razvoja aplikacij se platformske storitve uporabljajo tudi za številne druge namene [7]: sodelovanje med uporabniki, upravljanje s podatkovnim nivojem, kot orodja za merjenje in testiranje zmogljivosti programske opreme, shrambo podatkov, upravljanje s transakcijami

in posredniškimi sistemi. Za ta namen večina ponudnikov ponuja spremljanje postavljenih storitev. Še več, nekateri ponudniki omogočajo avtomatizirano okolje, ki se dinamično prilagaja glede na zahteve, ki prihajajo od zunaj. To pomeni, da bo okolje v primeru povečanja dostopov do aplikacije ali povečanja obremenitev sposobno postaviti novo instanco strežnika. V tem primeru govorimo o skaliranju storitev (angl. Scalability). Pri takšni realizaciji postavljanja platformskih storitev to pomeni postavljanje gruče strežnikov skupaj z izenačevalcem obremenitve.

2.1.3 Storitve programske opreme

Storitve programske opreme predstavljajo najvišji nivo storitev v oblaku. To pomeni, da je za izvedbo teh storitev potrebna uporaba vseh nivojev oblaka: od infrastrukturnega nivoja, preko nivoja platforme do nivoja končnih storitev. Pojav oblaka in s tem nove funkcionalnosti kot so skalabilnost, visoka razpoložljivost, večja stopnja integracije in večja varnost aplikacij predstavljajo odličen razlog za prehod aplikacij v oblak.

V skupino storitve programske opreme uvrščamo programsko opremo, ki jo ponudnik omogoča uporabniku ne glede na plačljivost. Primeri takšnih storitev, s katerimi se vsakodnevno srečujemo so Google Apps, Windows Live, Facebook, itd. Ponudniki teh storitev v celoti skrbijo za aplikacije ter za spodnji infrastrukturni nivo na katerega so aplikacije postavljene. Uporabnik dostopa do aplikacije preko odjemalca, ki je najpogosteje spletni brskalnik. Poleg tega ima občutek, da se aplikacije izvajajo lokalno, čeprav so postavljene na oddaljenem podatkovnem centru. Platforma oblaka, na katero je postavljena aplikacija, skrbi za izvedbo procesiranja, kot je na primer dostop do podatkovne baze, dostop do zunanjih aplikacij, itd. Storitve programske opreme so večinoma prilagodljive glede na nastavitve uporabnika, obstajajo pa tudi primeri kjer delujejo statično. Uporabniku torej ni treba skrbeti za vzdrževanje aplikacije, njihova odgovornost je omejena na upravljanje z lastnimi podatki znotraj aplikacije ali z aplikacijo.

2.2 Postavitveni modeli oblakov

Računalniški oblak lahko klasificiramo tudi glede na namen uporabe pri čemer obstajajo različni modeli oblakov glede na njihove značilnosti kot so na primer lokacija oblaka ter način upravljanja z infrastrukturo oblaka. Glede na definicijo NIST [8] je oblak lahko en izmed štirih definiranih modelov: zasebni, javni, skupnostni ali hibridni oblak.

2.2.1 Zasebni oblak

Pri podjetjih se zelo pogosto pojavljajo potrebe za racionalno izkoriščanje virov, ki jih imajo, oziroma ki jih najamejo. Pri zasebnem oblaku ima organizacija v svoji lasti zasebno infrastrukturo, iz katere dobi fizične ali virtualizirane vire. To pomeni, da organizacija izkorišča celotno procesorsko moč strežnika. Viri, ki jih organizacija uporablja, so lahko znotraj prostorov organizacije, na za to namenjeni strojni opremi, ki je v lasti podjetja. Druga možnost pri gradnji zasebnega oblaka je ponudba storitev zasebnega oblaka, tako da znotraj javnega oblaka izoliramo vire, ki so dodeljeni zasebnemu oblaku. Upravljanje in vzdrževanje fizične infrastrukture lahko prevzame organizacija ali pa ponudnik (v primeru gostujočega zasebnega oblaka). Ne glede na to ali je zasebni oblak postavljen v prostorih podjetja ali izven je značilnost slednjega večja varnost podatkov in večja zasebnost. To je predvsem zaradi dejstva, da so podatki dostopni le s strani enote in uporabnikov organizacije. Zaradi tega je zasebni oblak zelo uporaben v primeru kritičnih podatkov (na primer finančni zapisi).

2.2.2 Javni oblak

Javni oblak je postavitveni model oblaka, prilagojen za ponudbo računalniških virov kot storitev preko internetne povezave. Ponudniki javnega oblaka imajo v lasti infrastrukturo oblaka, ki jo nato dajo v najem ciljnim organizacijam ali posameznikom. Pri javnem oblaku ni nobene omejitve glede na to kdo ga uporablja, saj so storitve javnega oblaka javno dostopne. Razlika v primer-

javi z zasebnim oblakom je v deljenju virov infrastrukture med najemnike. To pomeni, da uporabniki javnega oblaka ne dobijo na razpolago celotne moči infrastrukture, ampak le deljene vire, ki so uporabniku na voljo glede na dogovor s ponudnikom - SLA. Ena izmed prednosti javnega oblaka je ta, da za vzdrževanje javne infrastrukture vedno skrbi ponudnik storitev. Z javnim oblakom so bili uvedeni pomembni koncepti računalništva v oblaku, kot je skalabilnost storitev in zaračunavanje glede na porabo (angl. Pay-As-You-Go).

2.2.3 Skupnostni oblak

Skupnostni oblak je uporaben kadar imamo več organizacij oziroma skupin uporabnikov z deljenim skupnim ciljem. Primeri takih organizacij so zdravstvene, vladne, znanstvene in podobne organizacije, ki potrebujejo skupne storitve programske opreme, platformske storitve ali celo infrastrukturne storitve. Skupnostni oblak ima podobne karakteristike kot zasebni oblak, vendar se razlikujeta v tem, da je skupnostni oblak namenjen za uporabo v skupnosti namesto v organizaciji. V primeru skupnostnega oblaka imamo več-najemniško infrastrukturo, ki je dodeljena skupnosti. Glede vzdrževanja obstajata dve možnosti. V prvem primeru imamo vzdrževanje s strani skupnosti - lahko s strani celotne skupnosti ali pa samo s strani posameznih organizacijah znotraj skupnosti. Druga možnost je vzdrževanje s strani ponudnika v primeru najema skupnostnega oblaka. Način zaračunavanja je podoben kot pri najemu storitev v javnem oblaku. Skupnostni model izkorišča prednosti javnega oblaka, kot je na primer več-najemniška infrastruktura, vendar je varnost podatkov veliko večja kot v javnem oblaku.

2.2.4 Hibridni oblak

Hibridni oblak je postavitveni model, ki združuje dva ali več omenjenih modelov oblaka. Ta tip oblaka je predvsem namenjen organizacijam, ki imajo postavljen zasebni oblak in potrebujejo hitro prilagajanje glede na spremembe

zahtev. Hibridni oblak zgradimo s sestavljanjem oblakov iz najmanj dveh infrastruktur, ne glede na tip oblaka (javni, zasebni, skupnostni). Čeprav obstajajo različne možnosti povezovanja je najpogostejša oblika povezovanja postavitev zasebnega oblaka z enim ali več javnimi oblaki. Pri tej postavitvi model izkorišča prednosti javnega in zasebnega oblaka.

Prednost javnega oblaka so navidezno neomenjeni viri, ki jih lahko dodajamo zasebnemu oblaku. To pride v poštev kadar je potrebno začasno povečati kapacitete zasebnega oblaka. To naredimo tako, da nekritične storitve prestavimo v javni oblak. Dodatna prednost je, da z delno uporabo javne infrastrukture podjetja prihranijo stroške za nakup dodatnih strežnikov. Po drugi strani pa je prednost zasebnega oblaka varnost podatkov, ki je v primeru hibridne postavitve seveda večja kot v javnem oblaku. Kar se tiče slabosti hibridnega oblaka lahko izpostavimo večjo ranljivost, saj se s povečanjem obsega postavitve odpira tudi večje območje za izvajanje napadov.

Poglavje 3

Avtomatizacija in orkestracija v oblaku

Avtomatizacija je zgodovinski pojem s katerim se srečujemo že vrsto let. Uporabljati se je začel v industriji, ko je bil narejen prvi proizvodni trak v tovarni Ford. Pomen avtomatizacije v industriji je uporaba strojev, ki jih upravlja kontrolni sistem, z namenom optimizacije proizvodnje preko izvajanja ponavljajočih se nalog. Danes je avtomatizacija prisotna na najrazličnejših področjih, tudi v IT industriji. Avtomatizacija v oblaku predstavlja sestavljanje nalog, ki jih je potrebno izvesti v enem avtomatiziranem procesu.

Razlogov za uporabo avtomatizacije v oblaku je veliko, našteli bomo le nekaj najpomembnejših primerov uporabe. Ponudniki se v produkcijskem okolju soočajo s potrebami upravljanja ogromnega števila virov. Avtomatizacija se nahaja na najnižjem - fizičnem nivoju. Kot primer lahko navedemo postavitev več podatkovnih centrov, konfiguracijo strežnikov za procesiranje in shrambo, nastavitev omrežnih komponent, itd. Avtomatizacija pomaga tudi pri izvajanju nalog, ki se pogosto ponavljajo. To se največkrat zgodi zaradi spremembe politik, ki jih moramo aplicirati večkrat, zato da obdržimo skladnost z regulativami.

Večina oblačnih platform ponuja več nivojev avtomatizacije, od zaporednega izvajanja operacij na podatkovnem centru oziroma na infrastrukturi

do postavljanja virtualnih strojev v oblaku. Za izvedbo avtomatizacije ima večina platform arhitekturo zgrajeno tako, da vsebuje komponento za centralizirano upravljanje virov. Komponente so sposobne upravljati orodja na nižjem nivoju (na primer orodja in knjižnice hipervizorja). Zelo pogosto imajo komponente že zgrajen uporabniški vmesnik, ki administratorju poenostavi upravljanje z virtualiziranim okoljem. Čeprav s temi komponentami dosežemo določen nivo avtomatizacije, to še ne pomeni avtomatizacije na nivoju oblaka. Ker oblak predstavlja abstrakcijo virov na višjem nivoju, so potrebne dodatne komponente za upravljanje oblaka. Te komponente izkoriščajo modularnost in nastavljivost procesov na nižjem nivoju in jih vključijo v višjenivojski proces. Primer takšnih procesov so postavitev virtualnega stroja na virtualnem podatkovnem centru, dodajanje predlog v katalog, upravljanje organizacij oblaka, itd.

Orkestracija po drugi strani predstavlja način kako združiti procese, ne glede na to kakšnega tipa so. Tukaj govorimo o kombinaciji tehničnih procesov (avtomatizacijske naloge) in poslovnih procesov (Information Technology Infrastructure Library - ITIL). To pomeni, da lahko vsak proces, ki smo ga nastavili in sprožili preko čarovnikov (angl. Wizards) s pomočjo omenjenih komponent oblaka, zapakiramo kot delovni tok in ga ponovno uporabimo. Orkestracija nam tudi omogoča, da kreirane delovne tokove in/ali procese povežemo v en proces. Na ta način, lahko orkestriramo delovanje celotnega okolja oblaka, ne glede na heterogenost infrastrukture. Nenazadnje pa je orkestracija način kako dosežemo pomembne zmožnosti in koncepte oblaka kot so avtomatizirano postavljanje storitev, avtomatizirano skaliranje, avtomatizirana izdelava varnostnih kopij ter avtomatizirano okrevanje po napaki.

Najpomembnejša prednost, ki jo orkestracija prinaša na področju računalništva v oblaku, je orkestracija storitev. Orkestracija je v tem primeru ustvarjanje zaporedja operacij, potrebnih za omogočanje storitev. Storitev je potrebno izvesti z enotnim procesom in brez ročnih operacij s strani ponudnika. Ne glede na to za kakšen tip storitev gre, z avtomatizacijo dosežemo pohitritev izvedbe storitev in večjo zanesljivost izvedbe postavitve.

Razlika med orkestracijo in avtomatizacijo je zelo pomembna, čeprav ponudniki ta dva procesa pogosto ne ločijo. Če pogledamo bolj podrobno: avtomatizacija predstavlja izvajanje specifičnih nalog, ki opravljajo posamezno operacijo, najpogosteje preko pisanja skript. Po drugi strani pa orkestracija združuje množico avtomatiziranih nalog in tako omogoča izvedbo celovitega procesa, ne glede na to ali je tehničnega ali poslovnega tipa.

3.1 Orodja za avtomatizacijo

Orodja za avtomatizacijo so se pojavila s potrebo po združevanju zapletenih ročnih poslov v enoten proces. Ta orodja lahko zajemajo različne procese, ki se izvajajo na infrastrukturi, od avtomatizacije fizičnih strežnikov, do postavitve programske opreme. Obstajajo različni tipi orodij glede na stopnjo razvoja funkcionalnosti. Dva najbolj razvita tipa orodij sta orodje za avtomatizirano upravljanje z infrastrukturo oblaka in orodje za konfiguracijo oblaka.

Razlika med omenjenima tipoma orodij je ta, da so orodja za konfiguracijo en nivo višje. To pomeni, da orodja za upravljanje z infrastrukturo za neposredno upravljanje z viri uporabljajo funkcionalnosti hipervizorja. Po drugi strani pa orodja za konfiguracijo uporabljajo funkcionalnosti infrastrukturnih orodij in omogočajo enotno točko avtomatizacije več različnih platform.

V nadaljevanju bomo podali kratek pregled najbolj uporabljenih orodij za izvedbo avtomatizacije.

3.1.1 Orodja za upravljanje z infrastrukturo oblaka

Na voljo so številna orodja, ki omogočajo upravljanje z infrastrukturo oblaka. Pri pregledu analiz odprtokodnih in komercialnih rešitev [9], [10] in [11] za upravljanje z infrastrukturo smo ugotovili, da se med najboljše rešitve uvrščajo OpenStack, Eucalyptus in Amazon Web Services. Omenjene rešitve so na kratko predstavljene v nadaljevanju.

OpenStack

Openstack [12] predstavlja odprtokodno ogrodje za gradnjo oblaka. Platforma je sestavljena iz komponente za računanje (Nova in Glance), upravljanje z omrežjem (Quantum) in shrambo (objektno shrambo - Swift in shrambo blokov - Cinder). Platforma podpira različne hipervizorje kot so KVM, Citrix XenServer, VMware ESX/ESXi in Microsoft Hyper-V.

Eucalyptus

Eucalypus [13] predstavlja ogrodje, ki temelji na platformi Linux. Arhitektura ogrodja je sestavljena iz več komponent: upravljanje - Cloud Controller, shramba podatkov - Walrus, upravljanje z gručo - Cluster Controller, blokovna shramba - Storage Controller, upravljanje z virtualnimi stroji - Node Controller, opcijski VMWare Broker za interakcijo z rešitvami VMware. Omogočeno je upravljanje naslednjih hipervizorjev: KVM, Xen in VMware ESX/ESXi direktno ali preko vCenter.

Amazon Elastic Compute Cloud

Amazon Web Services je nabor storitev, ki jih ponuja Amazon kot javni oblak. Amazon kot rešitev upravljanja z infrastrukturo ponuja storitev Elastic Compute Cloud (EC2). Slednja se lahko uporablja za gradnjo hibridnega oblaka in tako omogoča razširitev lastne infrastrukture. Storitev oblaka lahko avtomatizirano upravljamo preko oblaka Amazon (CloudFormation), ali preko rešitev za konfiguracijo oblaka (na primer Puppet in Chef), ki jih bomo omenili v nadaljevanju.

3.1.2 Orodja za konfiguracijo oblaka

Iz pregleda virov [14] in [15] smo ugotovili, da sta najboljši orodji za avtomatizacijo in konfiguracijo oblaka Puppet Labs in Opscode Chef, zaradi česar smo ju tudi podrobno opisali. Prednost teh dveh orodij je sposobnost upravljanja s široko množico infrastrukturnih rešitev in okolij.

Puppet Labs

Puppet [16] predstavlja orodje za avtomatizacijo in upravljanje z infrastrukturo oblaka. Rešitev omogoča pisanje lastnih skript za avtomatizacijo opravil v različnih okoljih. Zraven ponuja še upravljalni vmesnik za lažji pregled in upravljanje z viri. Orodje Puppet se lahko integrira z nekaterimi storitvami javnega oblaka kot so Amazon Elastic Compute Cloud in Google Compute Engine. Prav tako omogoča avtomatizacijo postavljanja odprtokodnih rešitev oblakov OpenStack in Eucaliptus Cloud.

Opscode Chef

Chef je rešitev podjetja Opscode, ki omogoča avtomatizacijo na več nivojih. Predstavlja možnost avtomatizacije neposrednih fizičnih strežnikov (angl. Bare Metal Provisioning). Chef se lahko integrira z različnimi hipervizorji preko knjižnice libvirt [17]. Rešitev omogoča tudi integracijo s platformo oblaka VMware vCloud in storitev Amazon AWS. Poleg upravljanja z infrastrukturo, imamo na voljo tudi zmožnost avtomatizacije spodajležečim nivojem PaaS. Kot primer uporabe lahko omenimo konfiguracijo spletnih strežnikov (na primer Apache Tomcat), povezovanja z izenačevalcem obremenitve, ali pa tudi konfiguracijo vozlišč podatkovne baze. Chef omogoča kreiranje predlog za avtomatizacijo (t.i. recepti), ki se nato lahko ponovno aplicirajo na naslednja vozlišča in na ta način je omogočena izvedba skaliranja.

3.2 Orodja za orkestracijo

Orodja za orkestracijo so nujna orodja platform v oblaku, še posebej pri zasebnih oblakih, saj omogočajo upravljanje s storitvami organizacij. Orodja omogočajo izvedbo orkestracije, oziroma nastavljanje vrstnega reda izvajanja IT procesov. Podpirajo različne tipe orkestracije, ki jih v grobem razdelimo na tehnične in poslovne orkestracije. Ko govorimo o tehnični orkestraciji, imamo v mislih upravljanje z infrastrukturo oblaka: sestavljanje različnih

avtomatizacijskih procesov, ki združujejo virtualizacijo virov, delo z strukturami postavljenega oblaka, itd. Po drugi strani pa poslovna orkestracija pomeni upravljanje z IT storitvami organizacije. Sem spada upravljanje z incidenti (angl. Incident Management), upravljanje s problemi (angl. Problem Management), upravljanje s spremembami (angl. Change Management), upravljanje s sredstvi (angl. Asset Mangement), itd.

3.2.1 Arhitektura orodja za orkestracijo in način delovanja

Arhitektura orodij za orkestracijo se razlikuje od produkta do produkta, saj je odvisna od postavljene infrastrukture oblaka, ki jo upravlja. Če povzamemo dobre prakse orodij za orkestracijo so sestavni deli arhitekture strežnik, ki izvaja orkestracijo in vsebuje pogon za orkestracijo (angl. Workflow Engine), ena ali več podatkovnih baz za shrambo delovnih tokov ter odjemalec za upravljanje z delovnimi tokovi, ki je lahko aplikacija ali pa spletni vmesnik.

Ključni del orodij za orkestracijo je orkestracijski pogon. Pogon se pogosto integrira z komponentami oblaka, ki so izpostavljene preko programskega uporabniškega vmesnika, zato, da je omogočen dostop. Pogon je tisti ključni del, ki pregleda in izvede kodo orkestracije. V samem izvajanju je sposoben komunicirati z različnimi spletnimi storitvami, prenašati podatke med eno in drugo storitvijo ter izvesti transformacijo podatkov.

Pomemben del orodja za orkestracijo je tudi zmožnost upravljanja z vtičniki. Vtičniki so del orkestracijske platforme, ki omogočajo integracijo z zunanjimi komponentami. Slednje so pri procesu orkestracije komponente za upravljanje z infrastrukturo, zunanjimi podatkovnimi bazami, imeniškimi storitvami, spletnimi storitvami, itd. Vtičniki ne vzpostavljajo le komunikacije med komponentami, ampak tudi uvozijo delovne tokove in aktivnosti za lažje upravljanje ter izvajajoče se objekte za komponente, politike, itd.

Vmesniki za upravljanje z orkestracijo se med različnimi orodji za orkestracijo razlikujejo. Nekateri orkestratorji omogočajo izvedbo orkestracije le s pisanjem programske kode. Drugi imajo poleg pisanja kode še grafični vme-

snik za lažje izdelovanje delovnih tokov po principu povleči in spusti (drag and drop). Koda, ki se pri tem generira, je skrita pred razvijalcem, vendar jo je možno tudi spreminjati.

Delovanje orodja za orkestracijo pri izdelovanju delovnih tokov lahko opišemo v naslednjih korakih:

1. Najprej s pomočjo orodja za orkestracijo kreiramo delovne tokove. Orodje nam ponuja možnost izdelovanja delovnih tokovih na dva načina. Najpogostejši način je s pomočjo grafičnih predlog, ki jih je možno vstaviti kot sestavni del drugih delovnih tokov pri izgradnji kompleksnega delovnega toka. Pri orodjih, ki ne ponujajo grafičnega vmesnika, so predloge delovnih tokov podane v obliki kode oziroma metod.

2. Preden se koda izvede je potrebno preverjanje pravilnosti kode in validacija vhodnih in izhodnih parametrov. Čeprav nekatera orodja ta korak izpustijo, je preverjanje potrebno zaradi pravilnejšega razvoja delovnih tokov.

3. Orkestrator izvede delovni tok. Pri tem koraku se uporabi delovni tok kot predloga iz katere naredi nova instanca oziroma posel z lastnimi vhodnimi parametri. Kreirana instanca se nato izvede s pomočjo pogona za izvedbo orkestracij.

4. Spremljanje izvajanja. Večina orodij za orkestracijo ponuja mehanizem za vpogled v izvajanje delovnega toka. Orodja omogočajo pogled vsakega izvedenega koraka in predstavljajo neke vrste razhroščevalnik izvajanja delovnih tokov.

5. Zadnji korak izvajanja je prikaz rezultatov izvedenega toka oziroma izhodnih parametrov toka. V primeru, da se je delovni tok uspešno zaključil, je na voljo rezultat izvajanja. V primeru, ko med izvajanjem pride do napake, se delovni tok ustavi in namesto izhodnih parametrov vrne izjemo.

3.2.2 Pregled obstoječih rešitev za orkestracijo

Kot smo že omenili, je orkestracija v oblaku način kako zagotoviti pohitritev izvajanja procesov in naprednih funkcionalnosti oblaka. Za izvedbo orkestracije obstaja veliko rešitev, ki lahko orkestrirajo lastno okolje oblaka in/ali okolje oblaka drugih proizvajalcev. Pregled rešitev za orkestracijo oblaka nam pomaga najti skupne značilnosti orkestracijskih orodij, obenem pa ponuja pregled upoštevanja najboljših praks pri orkestraciji. Za določitev zmožnosti orkestracije poslovnih procesov, smo izbrali najboljše komercialne rešitve za upravljanje zasebnega oblaka glede na analize Gartner [18], Forrester [19] in Webopedia [20]. Pri orodjih smo primerjali karakteristike vgrajenih komponent za orkestracijo. Najprej smo pri vsaki izmed rešitev opisali tehnologijo. Pregled produktov smo izvedli glede na kriterije, ki jih je mora izpolnjevati vsaka orkestracijska rešitev. Na koncu smo podali rezultate analize skupaj z ugotovitvami.

Cisco Process Orchestrator

Cisco Process Orchestrator je rešitev za orkestracijo platforme Cisco Intelligent Automation for Cloud - CIAC. Glavna lastnost orkestratorja je upravljanje s tehnologijami Cisco kot so Cisco Unified Computing System, Cisco Cloud Portal, Cloud Physical Server Automation, Cisco Network Services, itd. Poleg tega ponuja še veliko avtomatizacijskih paketov za zunanje tehnologije kot na primer sistemi SAP, podatkovne baze SQL, DB2 in Oracle, upravljanje z tehnologijami VMware ESX/ESXi in vCenter. Ključne funkcionalnosti Cisco Process Orchestratorja, ki so podane v dokumentaciji [21], so:

- Orodje za avtomatizacijo (angl. Automation Engine) - ponuja okolje za orkestracijo, ki omogoča gradnjo in zagon delovnih tokov.

- Vtičniki za integracijo z zunanjimi orodij - vtičniki omogočajo pobiranje podatkov iz tretjih komponent. Na ta način je omogočena podpora

za integracijo s komponentami za spremljanje, konfiguracijo in tudi povezavo s storitvenim centrom (angl. Service Desk).

- Vgrajen sistem za poročanje - poročanje po različnih modelih - donosnosti naložb (angl. Return on Investment - ROI) in revidiranje.
- Podpora za sporočanje - poročanje in opozarjanje o incidentih. Sistem nudi poročanje po elektronski pošti in pošiljanje poročil zunanjih platform preko integracije s storitvenim centrom ali platforme za upravljanje.

VMware vCenter Orchestrator

vCenter Orchestrator je orodje za orkestracijo platforme VMware, ki omogoča orkestracijo postavljenega oblaka vCloud. Komponenta je sestavni del oblaka VMware, s tem da se lahko integrira z VMware vCenter in tako omogoča neposredno upravljanje podatkovnega centra oziroma infrastrukture. Poleg zmožnosti integracije komponente vCenter Orchestrator s komponentami VMware (vCloud Director, vCenter Server, vCenter Chargeback, itd.) se lahko Orchestrator integrira tudi z zunanjimi sistemi. Podobno kot pri ostalih rešitvah za orkestracijo, se preko različnih vtičnikov povezuje s fizičnimi strežniki (UCS Manager) in z zunanjimi komponentami preko različnih protokolov (HTTP, HTTPS, SNMP, SOAP, REST). Rešitev omogoča grafično oblikovanje delovnih tokov, ki se izvajajo z uporabo vgrajenega skriptnega pogona Mozilla Rhino Engine. Delovni tokovi so dostopni na več načinov: dostop preko odjemalca vCenter Orchestrator, programskega vmesnika SOAP in spletnih storitev REST ter dostop preko spletnega vmesnika (angl. Frontend), ki se imenuje "Web-view". Orodja smo bolj podrobno opisali v poglavju 4.

Microsoft System Center Orchestrator

System Center Orchestrator je Microsoft-ova rešitev za orkestracijo [22], ki omogoča izdelavo in upravljanje z delovnimi tokovi oblaka Microsoft System

Center. Komponenta je sestavljena iz več strežnikov za različne namene (upravljanje, dostop do vmesnika, strežnik, podatkovna baza, itd.), ki predstavljajo sestavni del platforme oblaka Microsoft. Osnovna lastnost komponente orkestratorja je zmožnost integracije s komponentami System Center preko vtičnikov (t.i. integracijskih paketov), ki omogočajo uporabo funkcionalnosti posameznih komponent. Poleg tega se komponenta integrira tudi z ostalimi Microsoftovimi tehnologijami in z zunanjimi tehnologijami ter protokoli (Active Directory, Exchange, FTP, REST, komponente HP, VMware vSphere, itd.). Ključne lastnosti Microsoft System Center Orchestratorja:

- Spremljanje sistema - monitorji za izvajanja delovnih tokov.
- Zmožnost razhroščevanja (angl. Debugging) delovnih tokov za lažje odpravljanje napak.
- Obveščanje - preko različnih aktivnosti je omogočeno pošiljanje sistemskih in t.i. syslog obvestil.
- Zmožnost uporabe delovnih tokov preko spletne storitve REST.

BMC Atrium Orchestrator

BMC Atrium Orchestrator je še ena rešitev za združevanje procesov, ki deluje na različnih nivojih v okolju IT. Kot je opisano v [23], je Atrium Orchestrator sestavljen iz delovnih tokov in aktivnosti, ki upoštevajo standarde ITIL. Za izvedbo delovnih tokov je na voljo avtomatizacijsko okolje bazirano na jeziku BPEL (angl. Business Process Execution Language). Atrium je, podobno kot ostale rešitve za orkestracijo, mogoče integrirati preko vtičnikov z različnimi rešitvami za upravljanje s podatkovnim centrom. Na voljo je avtomatizirano upravljanje z produkti BMC (na primer BMC BladeLogic, BMC Remedy sistem) in drugimi tehnologijami kot sta NetApp, EMC. Poleg tega so no voljo tudi dodatni vtičniki za komunikacijo z zunanjim okoljem preko različnih protokolov (SMTP, HTTP in SNMP). Najpomembnejše lastnosti BMC Atrium Orchestratorja:

- Širok obseg operacij: postavljanje in konfiguracija, spremljanje in upravljanje, planiranje in urejanje, zahtevanje in podpora.

- Zmožnost upravljanja z incidenti - omogoča detekcijo in ukrepanje v primeru incidenta.

- Sinhronizacija dogodkov in podatkov med aplikacijami.

- Zmožnost upravljanja z življenjskimi cikli storitev in virov.

Amazon Simple Workflow Service

Amazon SWF [24] je orodje za orkestracijo procesov, ki se uporablja za upravljanje z javnimi storitvami oblaka Amazona. Za razvoj delovnih tokovih je na voljo ogrodje AWS Flow Framework. Slednje ponuja razvoj delovnih tokov, ki jim pravijo naloge (angl. Tasks) v programskem jeziku Java. Podobno kot pri ostalih orodjih za orkestracijo so sestavni del orkestracije delovni tokovi, ki kličejo aktivnosti. Delovni tokovi in aktivnosti se izvajajo v oblaku kot procesi (t.i. delavci) z virtualnimi stroji (angl. Virtual Machine Workers). Kot logiko za izvajanje delovnih tokov Amazon ponuja proces za odločanje (angl. Decider). Ta proces skrbi za optimizacijo izvajanja delovnih tokov, tako da sestavlja vrstni red izvajanja delovnih tokov in skrbi za prenos rezultatov izvajanja.

Funkcionalnosti orodja za orkestracijo Amazon SWF:

- Zapisi izvedenih delovnih tokov - ohranijo se koraki izvedbe delovnega toka, ter parametri s katerimi je bil tok zagnan. Pri tem je možna ponovitev izvedbe toka ali prenos parametrov iz izvedenega toka v drug delovni tok.

- Pregled in kontrola izvajanja - javanske funkcije omogočajo pregled stanja izvajajočega se toka, poleg tega je na voljo tudi zmožnost interakcije ter spreminjanje same konfiguracije izvedbe.

- Spremljanje izvajanja - pogled stanja izvajajočih se delovnih tokov, kar je mogoče z uporabo upravljalnega vmesnika Amazona - AWS Management Console.
- Obravnavanje napak oziroma izjem - z uporabo javanskih konstruktov, kot na primer try/catch/finally, je mogoče obravnavanje napak v posameznih delih kode.
- Dostop preko programskega vmesnika - zmožnost upravljanja z delovnimi tokovi z izvajanjem zahtev HTTP POST. Pri tem se za prenos podatkov v klicu REST uporablja format JSON.

3.2.3 Kriteriji za primerjavo

Za izdelavo primerjalne analize je bilo potrebno izbrati kriterije za primerjavo, ki smo jih uporabili pri ugotavljanju prednosti in pomanjkljivosti posameznih platform. Ker proizvajalci različno definirajo kriterije za najboljše prakse orkestracije, smo morali najprej ugotoviti katere so skupne lastnosti orodij za orkestracijo. Orodja za orkestracijo delno ali v celoti ponujajo zmožnosti orkestracije svojih lastnih platform (v primeru da obstajajo), pa ponujajo zmožnost orkestracije tudi ostalih platform. Pomembno je da večina orkestracijskih komponent vsebuje osnovne lastnosti za izvedbo orkestracije:

1. Omogočajo kreiranje delovnih tokov, ki so ponovno uporabni. To pomeni zmožnost postopnega nadgrajevanja delovnih tokov v bolj kompleksni tok. Poleg tega platforme za orkestracijo omogočajo povezovanje delovnih tokov in prenos parametrov med delovnimi tokovi.
2. Ponujajo orodje za izdelavo delovnih tokov. Nekatere platforme ponujajo grafični način izdelave delovnih tokov, ki je lažji za uporabo za razvijalca in tudi omogoča večjo preglednost. Pri ostalih platformah je orkestracija možna s pisanjem skript.
3. Omogočajo način zaganjanja delovnih tokov. Sistem je del platforme za orkestracijo in omogoča izvajanje kode delovnih tokov.

4. Spremljanje izvajanja delovnega toka. Večina platform za orkestracijo ponuja sistem za pregled delovanja posameznih delovnih tokov, nekateri omogočajo tudi razhroščevanje.

V nadaljevanju smo določili lastnosti, ki prinašajo dodane vrednosti orkestracijskih orodij in jih uvrstili v kriteriji za primerjavo. Čeprav je kriterijev za primerjavo veliko, smo izbrali le najpomembnejše lastnosti:

- Podpora tehnični orkestraciji - zmožnost upravljanja z viri, ki so dostopni v oblaku, upravljanje s komponentami oblaka, hipervizorji in fizično infrastrukturo.
- Podpora poslovni orkestraciji - upravljanje s poslovnimi procesi, incidenti, spremembami, problemi in podobnimi elementi, ki so del standarda ITIL.
- Podpora gradnikom za integracijo z zunanjimi sistemi oziroma heterogenost - predstavlja zmožnost izdelave delovnih tokov za upravljanje z infrastrukturo, ki pomeni identificiranje podprtih tipov platform oblaka oziroma hipervizorjev, na primer VMware vSphere, Microsoft Hyper-V ali odprtokodne rešitve oblaka.
- Uporabniška interakcija za realizacijo sistema za potrditev (angl. Approval System). Ta kriterij opisuje ali je vključena platforma za orkestracijo, ki vsebuje predloge delovnih tokov za pošiljanje zahtev sistemu za potrditev in spremljanje odgovorov teh zahtev.
- Obravnavanje napak (angl. Fault Handling) - sem spadajo zmožnosti zajemanja napak, prilagoditev izjem v primeru napak ter obveščanje. To je predvsem pomembno za realizacijo vrnitve sistema v prejšnje stanje.
- Podpora gradnikom za zagotavljanje orkestracije hibridnega oblaka - s tem kriterijem smo opisali zmožnosti izdelave delovnih tokov za migracijo in postavitev storitve iz zasebnega oziroma skupnostnega oblaka v javni oblak takrat, ko sta povezana v hibridni oblak.

- Integracija s sistemom za zaračunavanje (angl. Chargeback) - ali platforma vsebuje vključevanje procesa zaračunavanja porabe storitev kot sestavni del orkestracije storitev.

- Podpora uporabe vmesnikov REST ali SOAP - zmožnost dostopa zunanjih aplikacij do delovnih tokov. Delovni tokovi so v tem primeru izpostavljeni kot spletne storitve REST ali SOAP.

- Podpora orodjem CLI (angl. Command-Line Interface) - uporaba orodja z ukazno lupino, ki predstavljajo odjemalce za upravljanje z delovanjem orkestracijskih platform.

3.2.4 Rezultati primerjave

Tabela 3.1 prikazuje primerjavo orodij za orkestracijo po določenih kriterijih. Ugotovili smo, da orkestracijska orodja, ki smo jih primerjali, v celoti pokrivajo tehnično orkestracijo svojih lastnih platform, podpora ostalih platform pa je minimalna ali je sploh ni. Po drugi strani je podpora za poslovno orkestracijo možna le s platformama za orkestracijo Cisco in BMC. Podpora pri ostalih rešitvah, razen rešitve Amazon SWS, je omogočena preko dodatnih komponent. Pri tem se upravljanje izvaja na višjem nivoju. Vsi sistemi razen Amazon SWS podpirajo nastavitev vtičnikov za delo z različnimi zunanjimi sistemi, kar pomeni, da so komponente odprte za integracijo z različnimi sistemi, podatkovnimi bazami in protokoli. Iz tabele je razvidno, da je pri vseh produktih zagotovljena uporabniška integracija s sistemi za potrditev in da je omogočeno zajemanje napak. Največja pomanjkljivost orodij je nepodprtost gradnikov za zagotavljanje funkcij upravljanja hibridnega oblaka. Microsoft System Center Orchestrator je edino orodje, ki te funkcije omogoča. Kar se tiče upravljanja z zaračunavanjem je najbolj popolna platforma VMware, saj ponuja komponento za zaračunavanje vCenter Chargeback, ki je tudi vključena v proces orkestracije. Ostali produkti, ki omogočajo orkestracijo procesov zaračunavanja uporabe, ponujajo le osnovno zaračunavanje. Čeprav vsi produkti ponujajo dostop preko vmesnikov REST ali SOAP, se je ponovno

izkazal orkestrator platforme VMware z omogočenim dostopom preko obeh vmesnikov. Glede zmožnosti upravljanja orodij preko ukazno vrstičnih vmesnikov smo ugotovili, da nekatere platforme ponujajo lastno razvita orodja (SCO Job Runner, iCommand), vendar končne različice teh produktov ne vsebujejo (oziroma obstajajo le v beta različicah).

Glede na povedano lahko zaključimo, da so med orodji za orkestracijo najboljši VMware vCenter Orchestrator, Microsoft System Center Orchestrator in BMC Atrium Orchestrator. Če obravnavamo proces postavljanja storitev v oblaku je najbolj ustrezno orodje VMware vCenter Orchestrator in sicer predvsem zaradi vgrajene podpore integracije s sistemom za zaračunavanje. Pomanjkljivosti orodij, kot na primer podpora hibridnemu oblaku in poslovnim orkestracijam, so na platformi VMware realizirane s pomočjo dodatnih komponent vCloud Connector in vCloud Automation Center [25].

	Cisco	**VMware**	**Microsoft**	**BMC**	**AWS**
Orodja za orkestracijo	Process Orchestrator	vCenter Orchestrator	System Center Orchestrator	Atrium Orchestrator	Simple Workflow Service
Podpora tehnični orkestraciji	Da, podprte platforme: Cisco CIAC, VMware vSphere/ vCenter	Da, za platformo VMware vCloud	Da, za MS System Center, delno VMware vSphere	Da, za BMC Cloud Lifecycle Management, VMware vSphere	Da, za storitve v javnem oblaku Amazon AWS
Podpora poslovni orkestraciji	Change Management, Incident Management	Zagotovljeno preko vCloud Automation Center	Zagotovljeno s System Center Service Manager	Vgrajena podpora poslovne orkestracije	Ne podpira

Podpora gradnikom za integracijo z zunanjimi sistemi	Active Directory, BMC Remedy, IBM DB2, Microsoft SCOM, SNMP, VMware vCenter, vSphere, MSSQL Server, Oracle DB, SAP BWA, Email	Active Directory, AMQP, SQL, UCS Manager, spletne storitve REST in SOAP, SNMP, PowerShell, E-mail	Active Directory, HP iLO and OA, HP Operations Manager, HP Service Manager, IBM Tivoli Netcool/ OMNibus, VMware vSphere	Active Directory, NetApp, EMC, Oracle DB, DB2, SQL, XML	Ne. Podpora le za storitve AWS
Uporabniška interakcija s sistemom za potrditev	Podpira	Podpira, za skupine znotraj LDAP	Realizacija s Service Manager komponento	Podpira	Podpira human tasks
Obravnavanje napak (Fault Handling)	Vgrajeno znotraj elemente orkestracije	Podpora z elementi toka in skripte	Nastavitev znotraj delovnih tokov	Nastavitev znotraj delovnih tokov	Javanski konstrukti za obravnavanje napak
Podpora gradnikom za zagotavljanje hibridnega oblaka	Ne	Ne	Podpora za javnega oblaka Windows Azure	Ne. Podpora za AWS le preko BMC CLM	Ne

Podpora gradnikom za zaračunavanja uporabe	Ne	vCenter Chargeback vtičnika	Izvedba preko Service Manager vtičnika	Da, skupaj z BMC Cloud Lifecycle Management	Ne, ker gre za javni oblak
Podpora vmesnikov REST in SOAP	vmesnik SOAP	vmesnika REST in SOAP	vmesnik REST	vmesnik SOAP	vmesnik REST
Podpora orodjem CLI	Ne	Da, vendar z zunanjimi orodji	orodje SCO Job Runner	zagotovljeno z iCommand	Da, vendar z zunanjimi orodji

Tabela 3.1: Primerjava orkestracijskih orodij.

Poglavje 4

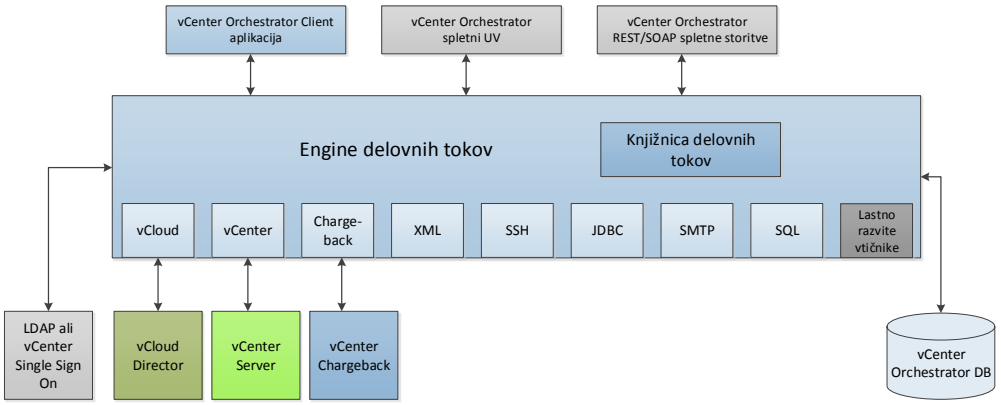
Opis VMware vCenter Orchestrator

VMware vCenter Orchestrator je komponenta za izvedbo tehnične orkestracije procesov oblaka VMware. Komponenta je del VMware vCloud Suite in jo je mogoče integrirati z večino komponent oblaka VMware. Za upravljanje komponente Orchestrator je na voljo odjemalec za vCenter Orchestrator, ki je tudi grafično orodje za izdelavo in upravljanje delovnih tokov. Poleg tega ima vgrajeno zmožnost nadzorovanja in upravljanja delovnih tokov preko spletne konzole komponente vCenter Server.

4.1 Arhitektura vCenter Orchestrator

Arhitektura komponente vCenter Orchestrator, ki je omenjena v viru [26], je organizirana s tremi osnovnimi moduli:

- Platforma vCenter Orchestrator - orodje s funkcionalnostjo razvijanja in zaganjanja poslovnih procesov.
- Knjižnica delovnih tokov - omogoča shranjevanje delovnih tokov v centralni repozitorij.



Slika 4.1: Arhitektura VMware vCenter Orchestratorja [26].

- Vtičniki - omogočajo povezavo med vCenter Orchestrator komponentami in zunanjimi komponentami, ki so lahko del oblaka VMware ali pa ne.

Arhitekturo komponente vCenter Orchestrator smo prikazali na sliki 4.1. V arhitekturi ima centralno vlogo pogon za orkestracijo, saj predstavlja osnovno komponento za izvedbo orkestracije. Orkestracijski pogon je tesno povezan s knjižnico delovnih tokov, kar predstavlja osnovni mehanizem za gradnjo in uporabo delovnih tokov. Za povezavo orkestracijskega pogona z zunanjimi sistemi, so k arhitekturi orkestratorja dodani vtičniki. Vtičniki omogočajo komunikacijo pogona za orkestracijo z ostalimi komponentami. Poleg tega lahko vtičniki vsebujejo predloge delovnih tokov, ki se dodajajo v knjižnico orkestratorja.

Poleg komponent, ki sestavljajo osnovno platformo za orkestracijo, je sestavni del orkestratorja tudi njegova podatkovna baza. V bazi se hranijo podatki o izvedenih instancah delovnih tokov, bolj podrobno - atributi delovnega toka. Platforma za orkestracijo je prav tako integrirana z datotečnimi storitvami (angl. Lightweight Directory Access Protocol - LDAP). Ena izmed zmožnosti LDAP integracije je Active Directory [27] za izvedbo avtentikacije uporabnika, ki upravlja z vCenter Orchestrator. Druga možnost pri izvedbi

avtentikacije je uporaba vCenter Single Sign On, kot sistema za posredovanje avtentikacije na VMware.
Glede načinov dostopa do pogona za orkestracijo, oziroma do delovnih tokov, imamo na voljo tri možnosti. Dostop je kot prvo mogoč preko odjemalca, ki ga predstavlja namizna aplikacija. Drugi način za dostop predstavlja spletni uporabniški vmesnik, ki služi le kot upravljalna konzola. Zadnji del arhitekture za orkestracijo predstavlja integracija s pomočjo spletnih storitev. To pomeni, da je omogočen programski dostop do delovnih tokov, ki so izpostavljeni kot spletne storitve tipa REST ali SOAP.

4.2 Pogoni vCenter Orchestratorja

VMware vCenter Orchestrator je sestavljen iz treh pogonov, ki skupaj omogočajo delovanje orkestracijske platforme: skriptni pogon, pogon delovnih tokov in pogon politik.

Skriptni pogon je osnovni pogon za izgradnjo blokov, ki sestavljajo delovne tokove. vCenter Orchestrator kot skriptni pogon uporablja Mozilla Rhino, ki je odprtokodna implementacija JavaScript in vključuje le jedro jezika JavaScript. To pomeni da implementacija ne vsebuje dele jezika JavaScript za upravljanje s dokumenti HTML. Pogon je sestavljen iz ukazne lupine za izvedbo skript, prevajalnika za pretvorbo kode JavaScript v javanske razrede in razhroščevalnika JavaScripta, kot je opisano v [28]. Pogon omogoča uporabo vseh funkcionalnosti JavaScript različice 1.7. Temu osnovnemu pogonu je dodanih še nekaj funkcionalnosti: kontrola različic, preverjanje tipa spremenljivk, upravljanje z imenskim prostorom in ravnanje z izjemami, kot je omejeno v [26].

Drugi pogon, ki je sestavni del platforme za orkestracijo, je pogon delovnih tokov. Ta pogon v ozadju uporablja skriptni pogon, da abstrahira sestavne dele kot so aktivnosti in delovni tokovi v objekte. Če pojasnimo, aktivnosti predstavljajo objekti sestavljeni iz kode JavaScript z dodanimi vhodi ter izhodom, ki predstavlja rezultat aktivnosti. Aktivnosti skupaj z

drugimi sestavnimi elementi pogona delovnih tokov kot so pogoji, zanke in izjeme združimo v delovni tok. Delovni tok predstavlja celoto, ki vsebuje število operacij za izgradnjo poslovnih IT procesov. Poleg tega ima pogon zmožnost upravljanja z novimi tipi objektov, ki jih uporabnik doda v vCenter Orchestrator preko različnih vtičnikov.

Pogon politik uporabljamo takrat, ko je potrebno zajeti podatke zunanjih komponent. Cilj pogona politik je proženje delovnih tokov glede na stanje ali interakcijo z zunanjimi objekti. Tukaj govorimo o spremljanju stanja objektov, ki so del VMware vCenter, kot so na primer virtualni stroji, katerih stanje se lahko spreminja. Pogon lahko zajame različnih metrike o opazovanem objektu in proži delovne tokove ter posamezne skripte. Prožilec je aktiven takrat, ko izmerjene vrednosti presežejo določeno mejo. Dodatna funkcionalnost pogona politik je interakcija z ostalimi komponentami preko protokolov AMQP in SNMP. S tem lahko komponente platforme oblaka VMware, kot sta na primer vCloud Director [29] in vCenter Operations Manager [30], prožijo politike s katerimi vCenter Orchestrator izboljša stanje infrastrukture.

4.3 Funkcionalnosti vCenter Orchestrator

V poglavjih 3.2.2 in 3.2.4 smo delno omenili nekatere funkcionalnosti platforme vCenter Orchestrator, ki smo jih uporabili za primerjavo z ostalimi produkti. V nadaljevanju bomo opisali ostale ključne funkcionalnosti platforme za orkestracijo:

- Centralizirano upravljanje - s pomočjo odjemalca imamo centralni pogled vseh gradnikov za izdelavo orkestracije in sicer od skripte, preko gradnikov za izdelavo delovnih tokov, vse do politike. To pomeni, da je na enem mestu omogočeno razvijanje, zaganjanje, spremljanje in beleženje različic (angl. Versioning) delovnih tokov.
- Validacija - platforma je opremljena z sistemom, ki pri razvijanju delovnih tokov preverja ali je tok veljaven - na primer veljavnost parametrov

in atributov, povezanost logičnih elementov, nastavitev izjem, itd. Validacija se avtomatsko izvede pred izvedbo delovnega toka, poleg tega pa jo je možno tudi ročno izvesti v katerikoli fazi razvoja.

- Beleženje različic - pomoč pri razvoju delovnih tokov je tudi zmožnost kontrole različic delovnih tokov. Sistem ohranja vse korake razvoja delovnega toka tako, da v bazo zapisuje vse različice delovnega toka. Nato je možno primerjanje delovnega toka s prejšnjimi različicami in povrnitev delovnega toka v katero izmed prejšnjih stanj.

- Trajna hramba informacij o delovnih tokovih - podatkovna baza ohranja podatke, ki so potrebni s strani vCenter Orchestrator. Primeri teh podatkov so informacije o vsakem koraku izvedbe delovnega toka, shranjevanje instance delovnega toka z vsemi atributi izvedbe, shranjevanje informacije o konfiguraciji, itd.

- Vmesnik Web 2.0 - spletne strani je možno zgraditi s pomočjo paketa Web View, ki je vgrajen v vCenter Orchestrator. Paket je sestavljen iz komponent za grajenje spletne strani in uporablja tehnologije: CSS, Ajax, orodje Dojo, komponente JWC iz ogrodja Trapestry, itd. Na ta način je mogoče, da uporabnik naredi vmesnik za dostop do funkcionalnosti vCenter Orchestratorja v obliki spletne strani.

- Dokumentiranje - platforma za orkestracijo omogoča izdelavo poročila oziroma dokumentacije v obliki PDF. Poročilo vključuje opis delovnega toka, opis različic, opis vseh vhodnih in izhodnih parametrov delovnega toka, atributov in izvorno kodo uporabljenih aktivnosti.

- Varnost - predvsem je poskrbljeno za varnost delovnih tokov in ostalih objektov, ki jih prenašamo v obliki paketov. Uporablja se PKI (Public Key Infrastructure) infrastruktura za digitalno podpisovanje paketov, DRM (Digital Rights Management) za dodeljevanje pravic pregledovanja vsebine izvoženih paketov, ter upravljanje s pravicami uporabnikov za dostop do delovnih tokov in objektov. Za povečanje varnosti pri

dostopu do delovnih tokov iz vmesnika pa se uporablja enkripcija SSL oziroma komunikacija HTTPS.

4.4 Vtičniki

Vtičniki so sestavni del komponente za orkestracijo in omogočajo povezovanje z zunanjimi komponentami. Za olajšanje dela z ostalimi komponentami so na voljo vgrajeni vtičniki, ki smo jih omenili v poglavju 3.2.2. Omogočen pa je tudi razvoj lastnih vtičnikov za integracijo s poljubnimi komponentami preko ogrodja vCenter Orchestrator Plug-In SDK.
Kot je omejeno v viru [31], so vtičniki sestavljeni iz javanskih razredov, XML definicijske datoteke in paketa delovnih tokov. Javanski razredi omogočajo definicijo osnovnih konstruktov vtičnika kot na primer: definicija vtičnika - Plug-In Adapter, tovarna za definiranje načina iskanja objektov - Plug-In Factory, upravljalci dogodkov - Event Handler, itd. Ker platforma za orkestracijo uporablja pogon JavaScript za izvajanje delovnih tokov, je potrebno ustrezno preslikati definirane objekte. Za ta namen se uporablja vso.xml definicijska datoteka, ki preslika javanske razrede, metode ali objekte v ustrezne gradnike komponente vCenter Orchestrator. Zadnji element vtičnikov so delovni tokovi, ki jih v skupini zapakiramo, in tako olajšamo nadaljnji razvoj delovnih tokov.

Poglavje 5

Izdelava delovnega toka za postavitev storitve v oblaku

Kot primer orkestracije storitev v oblaku smo izdelali delovni tok za postavitev storitve Apache Tomcat. Na ta način smo pokazali kako uporabiti tehnologijo VMware za postavitev storitev na zahtevo. Postopek postavljanja virtualne aplikacije je kompleksen proces sestavljen iz več korakov, ki jih bomo opisali v tem poglavju. Delovni tok smo razvili na modularni način in s tem omogočili ponovno uporabo istega toka, ki je lahko uporaben pri gradnji drugih storitev v oblaku ali za skaliranje obstoječih storitev. Primer postavitve storitve smo razvili na intuitivni način, tako da smo postopoma razvili procese, ki skupaj sestavljajo delovni tok. V razvojnem procesu smo najprej instancirali aplikacije ter pripravili parametre virtualne strojne opreme virtualnega stroja. Potem smo izvedli prilagoditev parametrov operacijskega sistema in izdelali prilagoditvene skripte za postavljanje strežnika. Dodatno smo razvili delovna tokova, ki omogočata ravnanje v primeru napak in realizacijo procesa razveljavitve postavitve storitve (angl. Rollback). Celotni postopek postavitve storitve v oblaku smo prikazali v poglavjih, ki sledijo.

5.1 Priprava virtualne namenske aplikacije

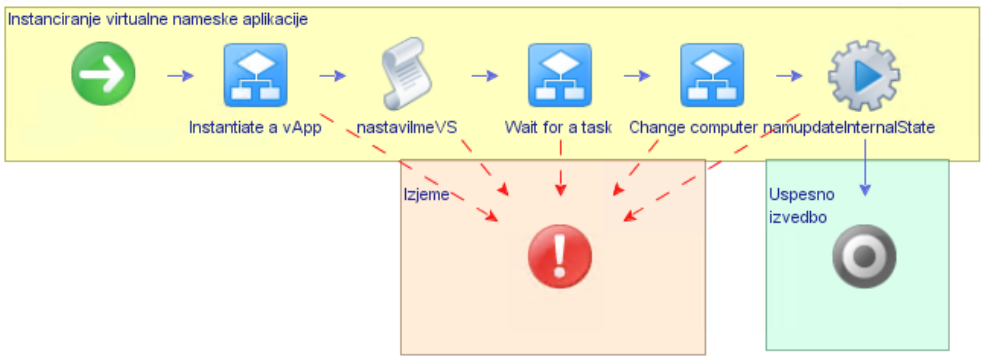
V prvi fazi postavitev storitve v oblaku smo pripravili storitve oziroma virtualne namenske aplikacije (angl. VApp). VMware združuje več virtualnih strojev v enoto imenovano virtualna namenska aplikacija. Storitev smo na začeteku pripravili v obliki predloge (angl. VApp Template) zato, da jo lahko izkoriščamo za instanciranje storitev. Predloga vsebuje konfiguracijske parametre zapisane v datoteki OVF in datoteko, ki predstavlja virtualni disk. V našem primeru, smo pripravili virtualne namenske aplikacije, ki vsebujejo virtualni stroj z nameščenim spletnim strežnikom Apache Tomcat. Za pripravo aplikacije smo izdelali delovne tokove za instanciranje aplikacije, za nastavitev parametrov virtualne strojne opreme in nastavitev omrežja.

5.1.1 Instanciranje virtualne namenske aplikacije

Instanciranje pomeni dodajanje nove virtualne namenske aplikacije v oblak, direktno iz predloge. Pri instanciranju se uporabljajo parametri za postavitev, ki so nastavljeni v datoteki OVF predloge. Instanciranje predstavlja začetni korak postavitve storitve, saj tukaj dobimo objekt tipa virtualne namenske aplikacije na katerem izvedemo konfiguracijo in prilagoditev (angl. Customization).

Za namen instanciranja smo izdelali delovni tok, ki naredi novo instanco iz izbrane predloge glede na nastavljene vhodne parametre. Delovni tok za instancianje smo podali na sliki 5.1.

Pri izdelavi delovnega toka smo uporabili obstoječe delovne tokove in aktivnosti iz vtičnika vCloud, ki pomagajo pri instanciranju virtualnih namenskih aplikacij. Za vključevanje instanciranja v procesu orkestracije vCenter Orchestrator ponuja že obstoječi delovni tok z imenom “Instantiate a vApp”. Delovni tok v ozadju sestavi in izvede klice vCloud API, ki izvedejo instanciranje. Za zagon delovnega toka je potrebno nastaviti vhodne parametre, ki poleg predloge virtualne aplikacije vključuje še: ime in opis nove virtualne aplikacije, virtualni podatkovni center na katerem se naj instanca postavi,



Slika 5.1: Delovni tok - instanciranje virtualne namenske aplikacije.

nastavitev, ali se aplikacija postavi in vklopi, čas najema za izvajanje storitev (angl. Runtime Lease Time), čas najema za shrambo (angl. Storage Lease Time), itd. V naslednjem koraku je potrebno nastaviti še ime virtualnega stroja, ker se pri instanciranju prevzame ime, ki je nastavljeno v predlogi. Na ta način preprečimo podvojitev imen, ki se lahko zgodi pri postavitvi storitve. Skripto za spremembo imena virtualnega stroja in posodobitev virtualnega stroja, smo podali v izvorni kodi 5.1. Po izvedbi kode delovni tok čaka, da se naloga posodabljanja izvede tako, da nastavi prožilec, ki se bo aktiviral ob odgovoru klica za posodobitev. Kot predzadnji korak, ki smo ga izvedli je sprememba imena operacijskega sistema (angl. Computer Name) virtualnemu stroju. Ime, ki ga tukaj nastavimo, se bo apliciralo pri izvedbi prilagoditev operacijskega sistema. Za izvedbo te operacije je na voljo delovni tok "Change computer name", ki je del knjižnice vCloud. Kot zadnji element delovnega toka smo dodali vgrajeno aktivnost "updateInternalStatevApp", ki omogoča posodobitev stanja virtualne namenske aplikacije. Na ta način smo dobili objekt virtualne namenske aplikacije, ki se bo nadalje uporabil kot vhod naslednjega delovnega toka.

Izvorna koda 5.1: Spreminjanje imena virtualnega stroja.

```
// pridobi prvi VS
var nizVirtualk = newVApp1.getChildrenVms();
```

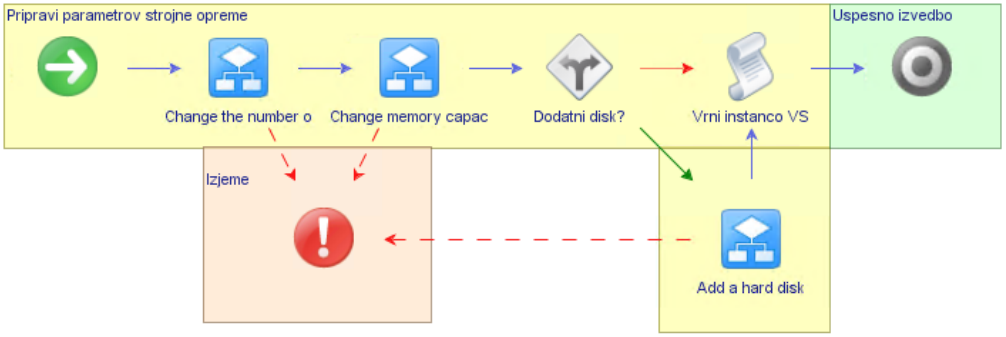
```
virtualniStroj = nizVirtualk[0];

// spremeni ime VS
virtualniStroj.name = imeVM;
updateNameTask = virtualniStroj.update();

// vrni VNA
instanciranVappAttr = newVApp1;

// vrni VS
vmSpremenjenoIme = virtualniStroj;
```

5.1.2 Priprava parametrov virtualne strojne opreme

Vsaki storitvi v oblaku je potrebno določiti tudi parametre virtualne strojne opreme, ki na VMware vCloud sodijo v "VirtualHardwareSection" del. Če povzamemo, so to parametri za konfiguracijo virtualnih procesorjev, virtualnega pomnilnika, konfiguracijo virtualnega diska in omrežne kartice za posamezni virtualni stroj, ki je del virtualne aplikacije. V tem delu smo opisali delovne tokove za nastavitev parametrov virtualne strojne opreme razen konfiguracije omrežne kartice, ki ga bomo opisali v poglavju 5.1.3.

Z namenom nastavitve virtualne strojne opreme virtualnemu stroju smo izdelali delovni tok, ki spremeni nastavitve virtualnega stroja. Delovni tok s katerim smo pripravili parametre virtualne strojne opreme je prikazan na sliki 5.2. Sestavni del delovnega toka so tudi predloge delovnih tokov, ki so del vtičnikov VMware za delo z vCloud Director in omogočajo spremembo virov virtualnega stroja.

Najprej smo za spremembo števila procesorjev uporabili in spremenili delovni tok "Change a number of CPUs", ki nastavi novo vrednost števila procesorjev. Pri tem smo spremenili parameter za alokacijo deležev virov na procesor (angl. Shares Per CPU) in nastavili normalno obnašanje. Ostalih



Slika 5.2: Delovni tok - priprava parametrov virtualne strojne opreme.

parametrov, kot je na primer hitrost procesorja, ne moremo nastaviti, ker so vnaprej določeni glede na izbran virtualni podatkovni center, kjer se storitev postavi.

V naslednjem koraku smo spremenili velikost virtualnega pomnilnika. Kot virtualni stroj, ki je vhodni parameter, smo nastavili posodobljeni objekt virtualnega stroja, kateremu smo prej nastavili novo število virtualnih procesorjev. V tem primeru smo uporabili delovni tok "Change memory capacity".

Tretji delovni tok za spremembo parametrov virtualne strojne opreme je delovni tok, ki dodaja dodatni disk virtualnemu stroju. Tukaj smo dodali možnost dodajanja novega virtualnega diska, ki se doda poleg obstoječih virtualnih diskov, ki so del predloge virtualne aplikacije. Tukaj smo uporabili delovni tok "Add a hard disk", ki izbranemu virtualnemu stroju doda disk. Kot vhodne parametre delovnega toka, smo nastavili velikost virtualnega diska v MB in tip diskovnega krmilnika.

Na koncu delovnega toka smo dodali skripto, ki kot izhodni parameter nastavi ustrezni objekt virtualnega stroja.

5.1.3 Priprava omrežja

Za izvedbo priprave omrežja s pomočjo vCenter Orchestrator se, podobno kot pri prejšnjih operacijah, uporablja vCloud vmesnik za dostop do operacij nad omrežjem oblaka vCloud. Za razumevanje delovanja delovnega toka je potrebno poznati osnovne konstrukte omrežja platforme vCloud, ki jih bomo podali v nadaljevanju. Platforma oblaka vCloud uporablja tri različne tipe omrežja: zunanja omrežja (angl. External Network), organizacijska omrežja virtualnih podatkovnih centrov (angl. Organization vDC Network) in omrežja virtualnih namenskih aplikacij (angl. vApp Network).

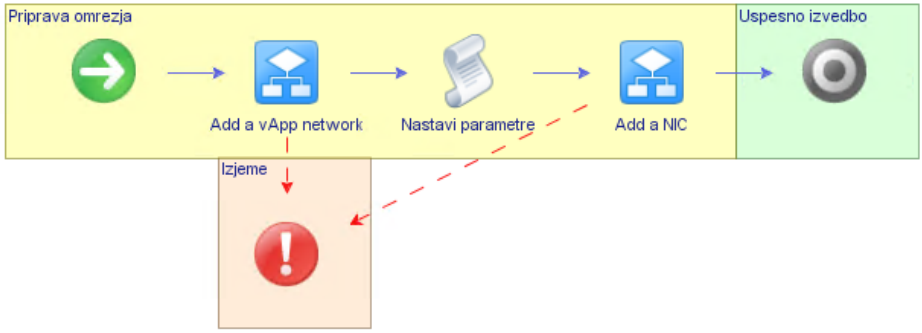
Zunanja omrežja so abstrakcija logičnih omrežij na nivoju vSphere, ki omogočajo povezavo med organizacijskimi omrežji. Poleg tega zunanja omrežja ponujajo dostop virtualne namenske aplikacije do interneta.

Drugi tip omrežij se uporablja s strani organizacije za medsebojno povezovanje virtualnih namenskih aplikacij znotraj organizacije. Obstajajo trije tipi organizacijskih omrežij: (1) zunanja organizacijska omrežja z direktno povezavo, ki omogočajo neposredno povezavo na zunanje omrežje, (2) zunanja organizacijska omrežja z povezavo NAT, namenjena organizaciji za dostop do zunanjega omrežja preko preslikave NAT naslovov IP in (3) notranja organizacijska omrežja, kjer je ves promet namenjen eni organizaciji in je izoliran od ostalih omrežij.

Omrežja virtualnih namenskih aplikacij omogočajo povezavo med virtualnimi stroji znotraj iste aplikacije. Ta tip omrežij lahko izolira promet znotraj virtualne namenske aplikacije ali pa se povezuje z organizacijskimi omrežji tako, da usmerja zunanji promet k tem omrežjem. Omrežja virtualne namenske aplikacije je potrebno v našem primeru nastaviti, saj je postavitev omrežja sestavni del storitev.

Za pripravo omrežja virtualnega stroja smo razvili delovni tok, ki nastavi omrežje virtualne namenske aplikacije in ga poveže z virtualnim strojem znotraj aplikacije. Shemo delovnega toka smo prikazali na sliki 5.3.

Delovni tok uporablja vgrajeni delovni tok “Add a vApp network”, ki kreira konfiguracijo omrežja virtualne namenske aplikacije glede na izbran način



Slika 5.3: Delovni tok - priprava parametrov omrežja.

povezovanja z organizacijskim omrežjem. Na vhodu delovnega toka moramo nastaviti določeno število parametrov, ki se razlikujejo glede na način povezovanja. Tu nastavimo parametre kot so ime omrežja, opis omrežja, omrežno masko, prehod (angl. Gateway), IP naslov DNS strežnikov, itd. Delovni tok smo dopolnili z lastno skripto, ki nastavi ime omrežja, v primeru, da smo izbrali direktno povezavo in tako odpravimo pomanjkljivost vgrajenega delovnega toka. Skripto smo prikazali v izvorni kodi 5.2.

Izvorna koda 5.2: Skripta za nastavitev imena omrežja.

```
if (modPovezovanja == VclFenceModeValuesType.BRIDGED) {
    imeVNAOmrezje == orgOmrezje.name;
}
imeVNAOmrezjeAtr = imeVNAOmrezje;
```

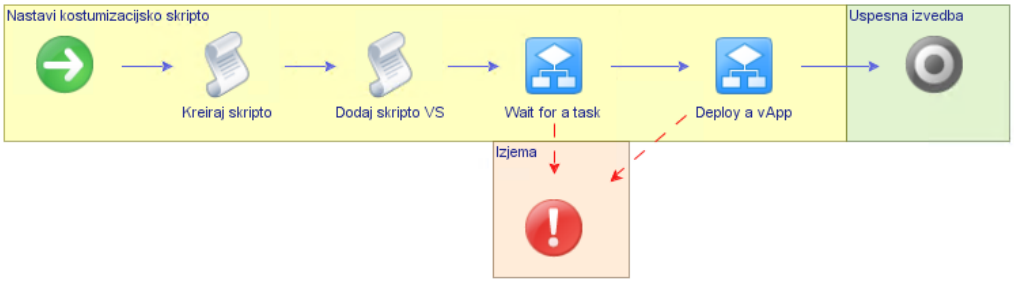
Zadnji element delovnega toka je vključen delovni tok “Add a NIC”, ki dodaja virtualno omrežno kartico izbranemu virtualnemu stroju. Za zagon tega toka je poleg izbire virtualnega stroja potrebno kot parameter podati tudi način dodeljevanja IP naslovov (angl. IP Addressing Mode) skupaj s parametri IP naslova, ki ga izberemo ob zagonu glavnega delovnega toka. Kot ciljno omrežje za povezavo izberemo omrežje, ki smo ga kreirali s prejšnjim delovnnim tokom.

5.2 Prilagoditev storitve

Ko govorimo o prilagoditvi storitve imamo v mislih izvajanje operacij znotraj virtualnega stroja, ki kot končni rezultat izvajanja ponujajo storitev spremenjeno po naši želji. Če povemo bolj podrobno - prilagoditev storitve lahko razdelimo na del prilagoditve operacijskega sistema nameščenega na virtualnem stroju in del prilagoditve storitev (aplikacijskega strežnika, spletnega strežnika, podatkovne baze, itd.). Za prilagoditev operacijskega sistema je v večini primerov poskrbljeno s strani platforme oblaka. Po drugi strani pa je potrebno za lastno prilagoditev storitev PaaS pripraviti prilagoditvene skripte.

Platforma VMware vCloud omogoča izvedbo prilagoditve storitev, ki se izvede pri postavljanju virtualne namenske aplikacije, oziroma pri zagonu posameznega virtualnega stroja. Prilagoditev se izvede le v primeru, ko je na virtualni stroj postavljena zbirka orodij VMware Tools. Vgrajena podpora za prilagajanje omogoča prilagoditev operacijskega sistema z naslednjimi zmožnostmi: sprememba administratorskega gesla za operacijski sistem, nastavitev spremembe gesla po prvi prijavi uporabnika/administratorja v sistem, zmožnost dodajanja virtualnega stroja v obstoječo domeno. Kot sestavni del prilagoditev platforma ponuja tudi zmožnost izvajanja lastnih prilagoditvenih skript.
Za namen prilagoditve smo z vCenter Orchestrator razvili delovni tok, ki orkestrira proces prilagajanja storitve. Shemo delovnega toka smo prikazali na sliki 5.4.

Kot sestavni del delovnega toka smo razvili skripte za prilagoditev strežnika Apache Tomcat. Skripte so namenjeni izvajanju na virtualnem stroju z operacijskim sistemom Linux in smo jih opisali v poglavju 5.2.1. Poleg tega smo opisali postopek izvajanja prilagoditev in podali javanske skripte za nastavitev prilagoditev v poglavju 5.2.2.



Slika 5.4: Delovni tok - prilagoditev virtualnega stroja.

5.2.1 Prilagoditvene skripte

Kot smo že omenili, so prilagoditvene skripte način nastavitev in sprememb parametrov storitev v oblaku. Platforma VMware vCloud nam omogoča izvajanje prilagoditvenih skript na različni stopnji prilagoditve. Glede na parametre ukazne vrstice lahko omogočimo izvajanje preden se prilagoditev operacijskega sistema virtualnega stroja začne ali pa potem, ko se ta izvede. Skripta za prilagoditev je lahko sestavljena iz največ 1500 znakov [29]. To pomeni, da je v večini primerov nemogoče izvesti celotno prilagoditev storitve, je pa dovolj za zagon prilagoditve. Zaradi teh omejitev, prilagoditev izvedemo tako, da pri pripravi predloge virtualnega stroja le-tega opremimo s potrebnimi skriptami za prilagoditve, ki jih potem zaženemo iz zagonske skripte.

Primer kode za kreiranje zagonske skripte za prilagoditev smo podali v izvorni kodi 5.3.

Izvorna koda 5.3: Kreiranje zagonske prilagoditvene skripte.

```
// Skripta ki se izvede po prilagoditvi OS
skriptaApache = '#!/bin/bash \n ' +
'if [ x$1 == x\"postcustomization\" ]; then \n ' +
'/opt/storitve/ApacheConfig ' +
vrata + ' ' + adminIme + ' ' + adminGeslo +
' >> /opt/storitve/InstallLog ' +
```

```
'\n fi ';
```

JavaScript koda omogoča kreiranje skripte ukazne lupine (angl. Shell Script). V primeru postavitve platformskih storitev je pomembno, da skripto izvedemo šele potem, ko se izvede prilagoditev operacijskega sistema. Slednje velja predvsem zaradi dejstva, da moramo najprej konfigurirati servise operacijskega sistema, nato pa lahko izvedemo dodatne prilagoditvene storitve. Zato smo v našem primeru uporabili izraz x$1 == x\"postcustomization\". Skripta najprej zažene skripto za prilagoditev, ki je shranjena znotraj virtualnega stroja. Kot parametre klica smo podali prilagoditvene parametre: vrata Apache strežnika ter ime administratorja in njegovo geslo. Na koncu izvedbe se izhod zapiše v datoteko za beleženje dogodkov. Omenjeno skripto za nastavitev parametrov znotraj virtualnega stroja smo opisali v izvorni kodi 5.4. Skripta spreminja konfiguracijske datoteke storitev Apache Tomcat in poskrbi za ponovni zagon storitve strežnika.

Izvorna koda 5.4: Postavljena skripta znotraj virtualnega stroja za prilagoditev strežnika Apache Tomcat.

```
#!/bin/bash

SERVER_CONF="/etc/tomcat6/tomcat6.conf"
SERVER_USERS="/etc/tomcat6/tomcat-users.xml"

# nastavitev vrat
sed -i -e "s/CONNECTOR_PORT=*/s/8080/$1/" $SERVER_CONF

# nastavitev parametrov za avtorizacijo
sed -i -e "s/USERADMIN/$2/" $SERVER_USERS
sed -i -e "s/ADMINPASSWORD/$3/" $SERVER_USERS

service tomcat6 restart
```

5.2.2 Nastavitev in izvajanje prilagoditev

Kot smo že omenili v poglavlju 5.2, so prilagoditvene skripte sestavni del prilagoditve storitve. Za nastavitev prilagoditvene skripte se uporablja vCloud klic, ki nastavi vrednosti parametrov znotraj sekcije za prilagoditev gostiteljskega operacijskega sistema.

Kot sestavni del delovnega toka na sliki 5.4 smo napisali skriptni element za nastavitev zagonske prilagoditvene skripte. Skripto za nastavitev smo podali v izvorni kodi 5.5.

Izvorna koda 5.5: Nastavitev zagonske prilagoditvene skripte.

```
// Pridobi prilagoditveni del virtualnega stroja
var kostumizacijskiDel = vhodniVS
.getGuestCustomizationSection();

// Nastavi skripto
kostumizacijskiDel.customizationScript = customScript;

// Cakaj na posodobitev
vCloudTask = vhodniVS.updateSection(kostumizacijskiDel);
posodobljenVS = vhodniVS;
```

Skripta najprej pridobi sekcija za prilagoditev iz virtualnega stroja in nato spreminja parameter, ki predstavlja prilagoditveno skripto. V končni fazi je potrebno posodobiti prilagoditveno sekcijo virtualnega stroja.

Prilagoditev, ki smo jo pripravili, lahko na VMware izvedemo na dva načina: z inicializacijsko prilagoditvijo, ki se izvede avtomatsko pri prvi postavitvi virtualne namenske aplikacije in v primeru, da ročno zahtevamo prilagoditev ob naslednjem zagonu virtualnega stroja. V našem primeru ni potrebna eksplicitna ročna nastavitev, ker se virtualni stroj postavlja prvič.

Za postavitev virtualne namenske aplikacije smo uporabili delovni tok "Deploy a vApp", ki postavi vse virtualne stroje, ki so del te aplikacije. Vhodne parametre smo nastavili tako, da se pri postavitvi virtualni stroji

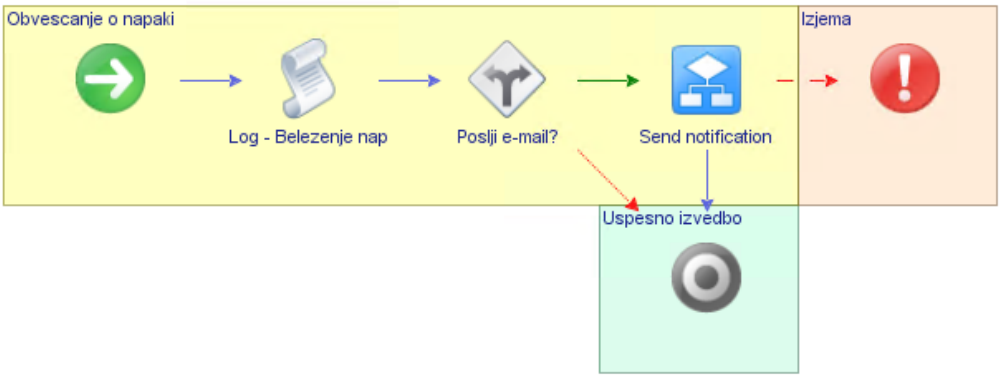
zaženejo in določili čas trajanja oziroma izvajanja storitev. Delovni tok za postavitev predstavlja zadnji element prilagoditvene sheme.

5.3 Obravnava napak in obveščanje

Obravnava napak pri orkestraciji pomeni zmožnost obravnave v primeru napake med delovanjem delovnega toka. Pri interakciji delovnega toka z zunanjimi spletnimi storitvami kot napako upoštevamo tudi podatke, ki so izven normalnih vrednosti. Ker mora biti orodje sposobno zajema napak, je na skriptni pogon vCenter Orchestratorja dodana še zmožnost obravnavanja napak.

Ob izdelovanju delovnih tokov je omogočen zajem in hranjenje izjeme oziroma napake v obliki tipa "String". Napako, v primeru da se pojavi, shranimo kot vmesni parameter delovnega toka, oziroma kot atribut delovnega toka. Platforma za orkestracijo VMware omogoča tudi propagiranje napak, kar nam omogoča, da zajete napake na nižjem nivoju prenesemo na višje nivoje vse do glavnega delovnega toka. Seveda je možno tudi preoblikovanje oziroma prepisovanje napake z lastnimi napakami, kar omogoča lažje odpravljanje napak.

Shema delovnega toka, ki omogoča obravnavanje napak je prikazana na sliki 5.5. Delovni tok kot vhodne parametre sprejema tip napake, vsebino napake in parameter, ki pove, ali se izvede tudi obveščanje po elektronski pošti. S pomočjo skriptnega elementa smo naredili zapis tipa in vsebine napake v datoteko za beleženje znotraj vCenter Orchestrator strežnika. Pri tem smo uporabili funkcijo System.error(), za nastavitev zapisa kot napako. Za izvedbo obveščanja smo uporabili vgrajeni delovni tok "Send notification", ki pošilja elektronska sporočila. Poleg tega smo za zagon tega delovnega toka nastavili specifične parametre SMTP strežnika kot so: IP naslov SMTP strežnika, SMTP vrata, uporabniško ime in geslo za račun prejemnika, naslov pošiljatelja, itd.



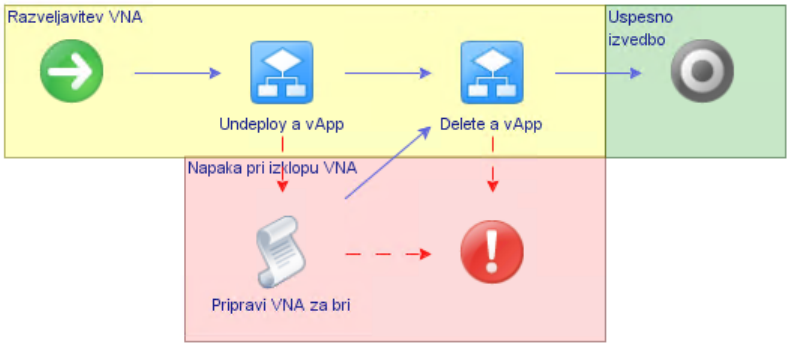
Slika 5.5: Delovni tok - obravnavanje napak in obveščanje.

5.4 Razveljavitev postavitve

V primeru, da imamo delovni tok namenjen izvedbi v produkcijskem okolju, ga moramo opremiti tudi z avtomatiziranim načinom razveljavitve postavitve. Razveljavitev je nujno potrebna, ker se pri izvajanju delovnega toka, ki na primer postavi neko storitev, lahko zgodi napaka v postopku izvedbe. V tem primeru se alocirajo določeni viri v oblaku (na primer zasede se del diska, kreira se omrežje, itd.), ki jih platforma za orkestracijo ob napaki ne more samodejno sprostiti, zato je potrebno narediti delovni tok, ki bo sprostil alocirane vire in tako razveljavil postavitev.

Pri postavljanju storitev na VMware sta na voljo dva tipa postavitve: postavitev storitve v obliki nove virtualne namenske aplikacije in postavitev storitve v obliki virtualnega stroja, ki ga združimo z obstoječo virtualno namensko aplikacijo. V prvem primeru imamo zelo enostaven način izvedbe razveljavitve storitve, tako da izvedemo vCloud operacijo, ki izbriše virtualno namensko aplikacijo. Večje težave se pojavljajo pri dodajanju virtualnega stroja v obstoječo virtualno namensko aplikacijo. V tem primeru je potrebno ugotoviti katere vire nova storitev uporablja in jih izbrisati samo v primeru povzročanja konfliktov z ostalimi deli virtualne namenske aplikacije.

V našem primeru postavitve virtualne namenske aplikacije smo storitve



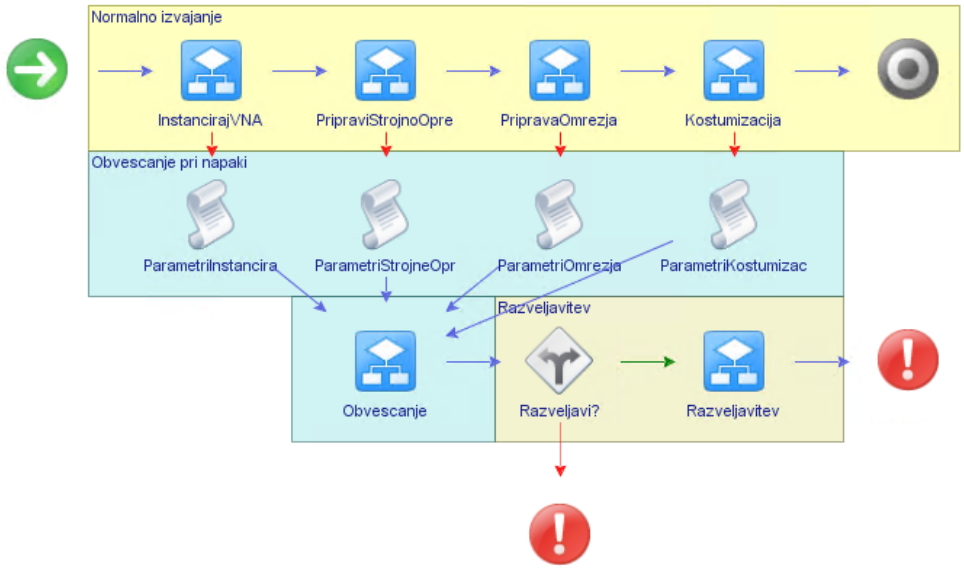
Slika 5.6: Delovni tok - razveljavitev postavljanja virtualne namenske aplikacije.

zgradili tako, da se vedno postavi nova aplikacija. Zaradi tega, smo pri procesu razveljavitve postavitve izvedli razveljavitev na nivoju virtualne namenske aplikacije.

Shema na sliki 5.6 opisuje delovni tok, ki smo ga razvili za razveljavitev postavitve. Prvi element delovnega toka je vgrajeni delovni tok "Undeploy a vApp", s katerem opravimo vCloud klic za zaustavitev virtualne namenske aplikacije. Do izjeme lahko pride v primeru, da je virtualna namenska aplikacija že izklopljena. Ne glede na izjemo, delovni tok poskuša izbrisati aplikacijo s pomočjo vgrajenega delovnega toka "Delete a vApp". Tok deluje tako, da pobriše celotno vsebino virtualne namenske aplikacije, skupaj z vsemi virtualnimi stroji in omrežji.

5.5 Delovni tok za postavitev virtualne namenske aplikacije

Delovni tokovi, ki smo jih do sedaj opisali, omogočajo le izvedbo posamezne operacije. Zato, da smo v enem delovnem toku omogočili postavitev aplikacije, smo združili in ustrezno konfigurirali vse omenjene delovne tokove. Delovni tok za postavljanje storitve je podan s shemo na sliki 5.7.



Slika 5.7: Delovni tok - postavitev virtualne namenske aplikacije.

Pri normalnem poteku delovnega toka se izvedejo zaporedne operacije: instanciranje, priprava virtualne strojne opreme ter omrežja in prilagoditev storitev. V primeru izjem pri delovanju smo omogočili nastavitev parametrov z informacijami o napaki ter ustrezno obravnavanje napak in obveščanje. Na koncu smo uporabili delovni tok, ki razveljavi postavitev aplikacije samo v primeru, ko je to potrebno, oziroma v primeru, ko obstaja objekt virtualne namenske aplikacije.

Poglavje 6

Izdelava delovnih tokov za avtomatizacijo storitev v hibridnem oblaku

Hibridni oblak je model računalništva v oblaku, ki je, čeprav ponuja veliko prednosti, v primerjavi z ostalimi modeli na nižji stopnji razvoja. To pomeni, da platforme, ki omogočajo upravljanje z zasebnim ali javnim oblakom, ponujajo le osnovne funkcionalnosti hibridnega oblaka. Če se osredotočimo na platformo VMware vCloud, ta kot sestavni del vsebuje dodatno komponento za vzpostavitev hibridnega oblaka - vCloud Connector [32]. Ta je sestavljena iz komponente strežnika - vCC server, ki se postavi v zasebnem delu hibridnega oblaka, ter vCC vozlišča, ki se postavi znotraj zasebne in javne postavitve oblaka. Čeprav je arhitektura komponente vCloud Connector dobro zasnovana, je komponenta še vedno v razvoju, tako da v različici 2.0 ne ponuja možnosti upravljanja z obremenitvami preko vmesnika API. Dodamo lahko še to, da komponenta ne pokriva naprednih funkcionalnosti, ki so kritične za oblak, kot je na primer dinamična postavitev nove obremenitve, ali optimizacija hibridnega oblaka. Zato smo v našem primeru razvili lastno implementacijo hibridne platforme, v kateri smo upoštevali način delovanja platforme vCloud Connector. Poleg tega smo izdelali delovne tokove

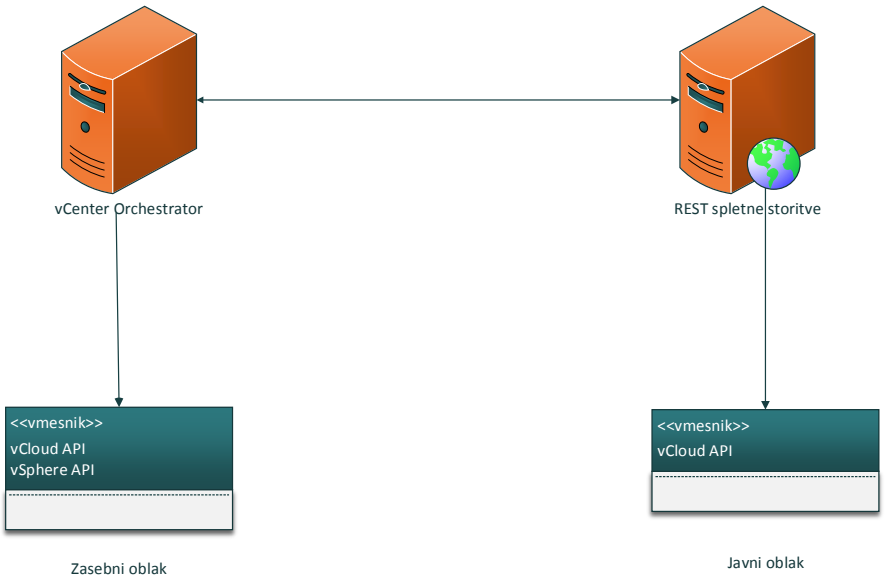
za omogočanje osnovne funkcionalnosti hibridnega oblaka, kot so migracija storitev, dinamična postavitev nove obremenitve in optimizacija virov med postavitvijo oblaka.

6.1 Implementacija hibridne platforme

Za izvedbo implementacije hibridne platforme smo uporabili podoben način komunikacije kot pri komponenti vCloud Connector. V ta namen smo na strani javnega oblaka postavili spletne storitve, ki se integrirajo z vCenter Orchestrator komponento. Spletne storitve izvedejo API klice vmesnikov VMware (največkrat vCloud API klice) zato, da se na strani javnega oblaka, ki temelji na VMware, izvedejo potrebne operacije. Te operacije smo razvili v programskem jeziku Java, z uporabo ogrodja vCloud SDK. Glede izpostavitve spletne storitve smo imeli na voljo tako REST, kot tudi storitev SOAP, saj je z vCenter Orchestrator mogoče upravljati z obema tipoma storitev. Zaradi manjše kompleksnosti in večje fleksibilnosti smo uporabili REST tip storitev. Na sliki 6.1 smo opisali implementacijsko arhitekturo hibridne platforme. Za realizacijo hibridnega oblaka sta ključni komponenti vCenter Orchestrator ter spletne storitve REST. Arhitekturo je mogoče poljubno nadgraditi. Kot primer nadgradnje bi predstavljal spletni uporabniški vmesnik za upravljanje z delovnimi tokovi komponente vCenter Orchestrator in s tem upravljanje z operacijami hibridnega oblaka.

6.2 Dinamična postavitev nove obremenitve

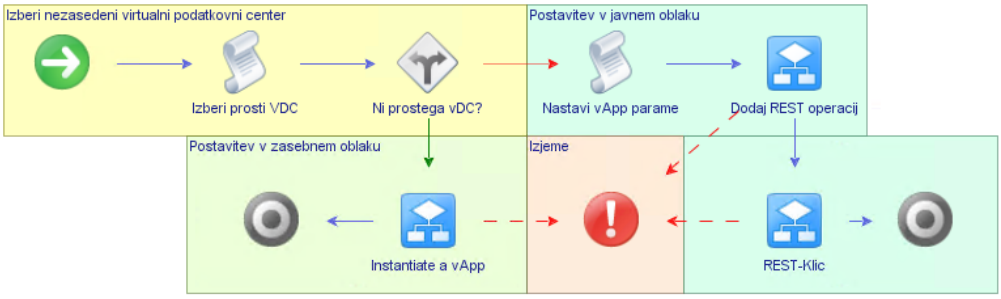
Pri dinamični postavitvi novih obremenitev je potrebno doseči zmanjšanje obremenitve celotne postavitve hibridnega oblaka. Pri tem želimo ohraniti zmogljivost zasebnega oblaka in obenem izkoristiti neomejene vire javnega oblaka. Zasebni oblak VMware vCloud omogoča postavitev organizacije znotraj enega ali več virtualnih podatkovnih centrov ponudnika (angl. Provider VDC). To pomeni, da so meje zasebnega oblaka nastavljene z konfiguracijo



Slika 6.1: Arhitektura implementacije hibridne platforme.

vCloud organizacije. Organizacija lahko upravlja z več organizacijskimi virtualnimi podatkovnimi centri, ki predstavljajo vire, kateri sestavljajo zasebni oblak. Vsakega izmed podatkovnih centrov lahko omejimo glede različnih kvot kot so: število omrežij in omrežnih vmesnikov, število virtualnih strojev, maksimalno število alociranih virov iz fizičnega podatkovnega centra, itd. Za izvedbo dinamične postavitve smo v našem primeru upoštevali izkoriščenosti virov organizacijskega podatkovnega centra. Pri vsakem od podatkovnih centrov smo upoštevali tri parametre in sicer izkoriščenost procesorskih enot, zasedenost pomnilnika ter zasedenost komponent za shrambo. Za vsakega izmed naštetih parametrov smo izbrali zgornjo mejo glede na razpoložljivosti virov.

Delovni tok za dinamično postavitev nove virtualne namenske aplikacije smo podali na shemi, ki jo prikazuje slika 6.2. V prvem koraku, smo preverili



Slika 6.2: Delovni tok - dinamična postavitev obremenitve.

obremenitev zasebnega oblaka tako, da smo poiskali organizacijski virtualni podatkovni center, ki ni zaseden. Skripto za povpraševanje po zasedenosti smo podali v izvorni kodi 6.1. V primeru da obstajajo neizkoriščeni viri znotraj zasebnega oblaka, za postavitev virtualne namenske aplikacije uporabimo izbrani prosti virtualni podatkovni center.

Izvorna koda 6.1: Nastavitev zagonske prilagoditvene skripte.

```
var organizacijskiVPC = vhodnaOrganizacija.getVdcs();
var steviloVDC = organizacijskiVPC.length;
var izbraniVDC = null;
for (var i=0; i < steviloVDC; i++) {
  var trenutniOVPC = organizacijskiVPC[i];
  var kapacitetaOVPC = trenutniOVPC.computeCapacity;
  var kapacitetaCPU = kapacitetaOVPC.cpu;
  var kapacitetaRAM = kapacitetaOVPC.memory;
  var kapacitetaShramba = trenutniOVPC.storageCapacity;
  var izkoriscenostCPU = kapacitetaCPU.used
                       / kapacitetaCPU.limit;
  var izkoriscenostRAM = kapacitetaRAM.used
                       / kapacitetaRAM.limit;
  var izkoriscenostShramba = kapacitetaShramba.used
                           / kapacitetaShramba.limit;
```

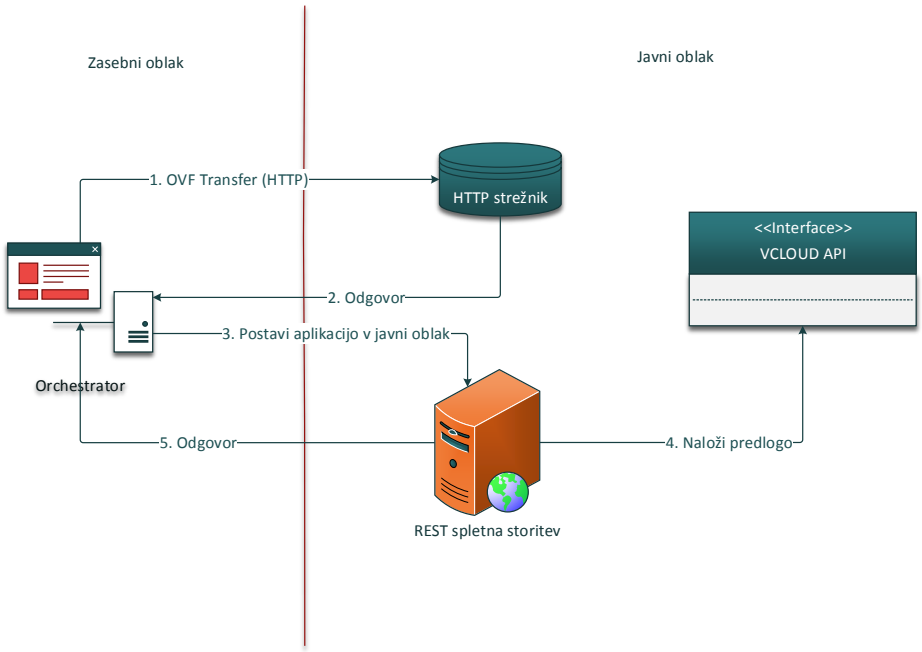
```
  if( izkoriscenostCPU < 0.80
        && izkoriscenostRAM < 0.85
        && izkoriscenostShramba < 0.70){
    izbraniVDC = trenutniOVPC;
    break;
  }
}
orgVDC = izbraniVDC;
```

V primeru, da prosti organizacijski virtualni podatkovni center ne obstaja, storitev postavimo v javni oblak. V ta namen imamo pripravljeno predlogo storitve znotraj javnega oblaka. Delovni tok pri postavitvi storitev v javni oblak izvede REST klic s katerim instancira storitev znotraj javne postavitve oblaka.

6.3 Migracija storitev iz zasebnega oblaka v javni oblak

Migracija storitev v hibridnem oblaku je proces prenosa delovnih obremenitev iz enega dela hibridnega oblaka v drugi. Migracija se največkrat uporablja takrat, ko prestavljamo nekritične storitve iz zasebnega v javni oblak. To je predvsem zaradi dejstva, da so podatki v javnem oblaku manj nadzorovani s strani same organizacije in posledično je varnost manjša, zato je bolje premestiti tiste storitve, ki ne vsebujejo kritičnih podatkov.

Za prenos podatkov o virtualnih strojih smo uporabili odprti virtualizacijski format (angl. Open Virtulization Format - OVF). OVF predstavlja standard, ki opisuje nastavitve virtualnega stroja v obliki XML. Poleg prenosa deskriptorja OVF, je nujno premestiti tudi virtualni disk virtualnega stroja, ki je zapakiran v eni ali več datotekah s končnico .vmdk. Slika 6.3 prikazuje realizacijo postopka migracije v hibridnem oblaku. Potek izvajanja operacij je naslednji:



Slika 6.3: Postopek migracije iz zasebnega oblaka v javnem oblaku.

1. V prvem koraku izvozimo virtualni stroj na deljeno HTTP lokacijo. Za realizacijo izvoza smo izkoristili zmožnosti vtičnika OVF Transfer, ki omogoča prenos virtualnih strojev iz ene vCenter postavitve v drugo. Pri tem lahko uporablja različne protokole za prenos podatkov kot na primer HTTP, HTTPS ali FTP. Pri naši izvedbi smo uporabili aktivnost exportOVF, ki je del vtičnika in zgradili delovni tok za izvoz virtualnega stroja. Aktivnost smo nastavili za prenos na deljeno lokacijo preko protokola HTTP.

2. Aktivnost delovnega toka spremlja odgovor o uspešni izvedbi prenosa.

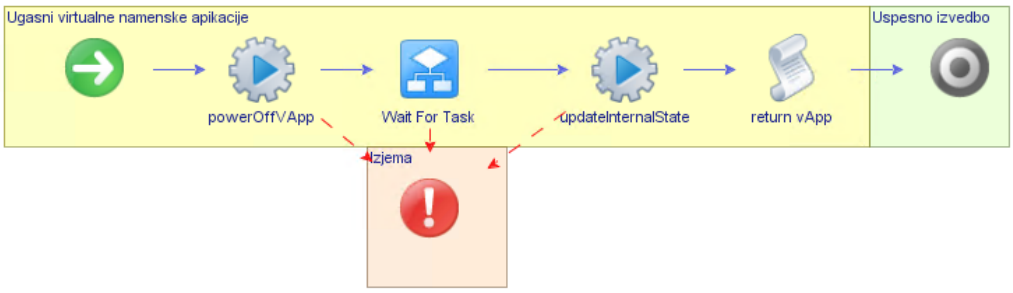
3. Izvedemo klic spletne storitve REST s pomočjo vgrajenega vtičnika REST za vCenter Orchestrator. Spletno storitev pokličemo skupaj s parametri o lokaciji virtualnega stroja ter parametri za postavitev v javni oblak.

4. Spletna storitev REST poskrbi za celotno postavitev virtualnega stroja v javni oblak. Storitev REST uporablja vmesnik javnega oblaka vCloud in izvede zaporedje operacij za postavitev kamor spadajo: uvoz virtualnega stroja v predlogo, kreiranje instance iz predloge, nastavitev parametrov za izvajanje ter zagon storitve.

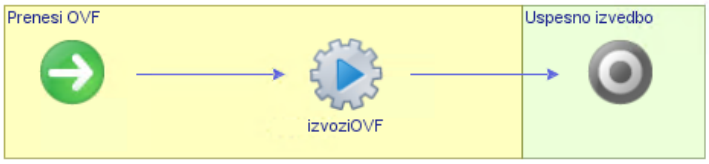
Opisani postopek migracije v našem primeru velja le za posamezni virtualni stroj, ki je del aplikacije. V primeru, da je storitev sestavljena iz več virtualnih strojev, je potrebno ponoviti prenos za vsak virtualni stroj posebej.

6.3.1 Izdelava delovnih tokov za izvedbo migracije v oblaku

Za izvedbo migracije smo na podlagi načrtovane arhitekture v poglavju 6.3 izdelali delovni tok, ki prestavi storitev iz zasebnega v javni oblak. Delovni tok smo zgradili iz naslednjih delovnih tokov in aktivnosti:



Slika 6.4: Delovni tok - ugasni virtualne namenske aplikacije.



Slika 6.5: Delovni tok - prenos datotek OVF.

1. Ugasni vApp – kot predpogoj za izvedbo migracije je potrebno, da je omenjena storitev pred prenosom izklopljena. Uporabili smo torej delovni tok, ki ustavi delovanje virtualne namenske aplikacije z izklopom vseh virtualnih strojev. Shema delovnega toka je podana na sliki 6.4.

2. Prenesi OVF - delovni tok uporabljamo za prenos virtualnega stroja na deljeno lokacijo preko protokola HTTP. Kot centralni sestavni del toka smo uporabili aktivnost exportOVF iz knjižnice OVF Transfer. Na vhodu delovnega toka je potrebno nastaviti objekt virtualnega stroja, ki ga prenašamo ter naslov HTTP.

3. REST-Klic - namen delovnega toka je izvajanje REST klica, ki omogoča nadaljevanje izvajanja migracije na strani javnega oblaka. Komponenta za orkestracijo preko vtičnika HTTP-REST izvede klic izpostavljene spletne storitve REST. Slika 6.6 predstavlja shemo delovnega toka za izvedbo klica spletne storitve. V nadaljevanju smo v izvorni kodi 6.2

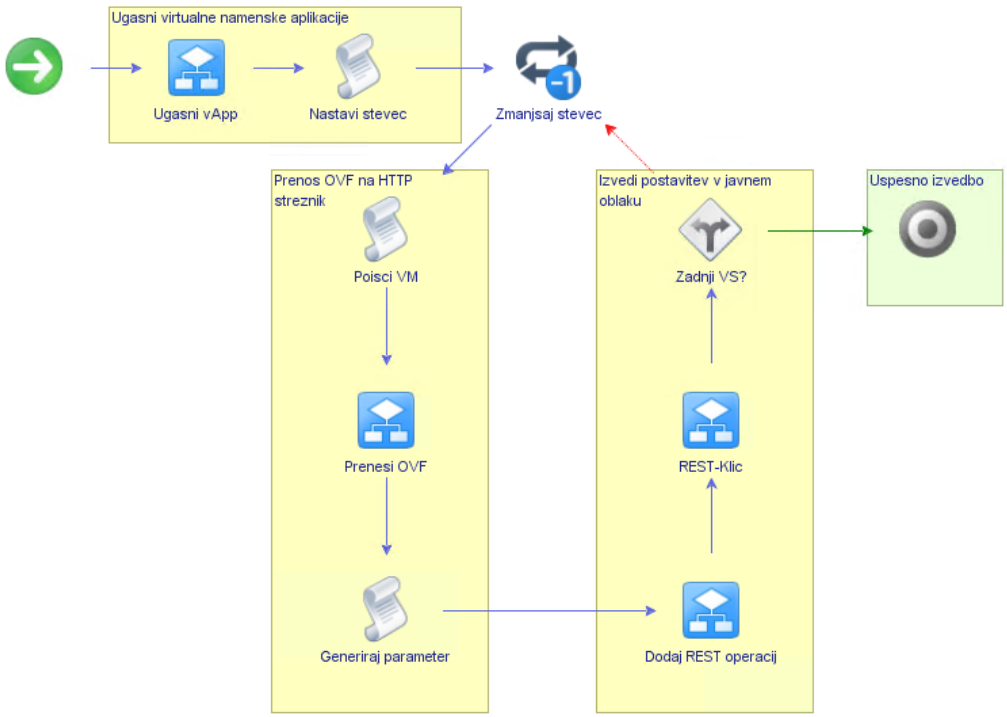
Slika 6.6: Delovni tok - klic spletne storitve REST.

podali tudi skripto, s katero dinamično generiramo REST klic ob izvedbi delovnega toka.

Izvorna koda 6.2: Vsebina skripte za REST klic.

```
var vhodniVrednostiParametrov = [];
var zahteva = restOperacija
.createRequest(vhodniVrednostiParametrov,vsebina);

//Nastavi zahtevo
zahteva.contentType = defaultContentType;
zahteva.setHeader("Accept",
        "application/*+xml;version=5.1");

// Nastavi odgovor
var odgovor = zahteva.execute();
status = odgovor.statusCode;
statusIzhod = status;
contentLength = odgovor.contentLength;
zaglavja = odgovor.getAllHeaders();
vsebinaOdgovora = odgovor.contentAsString;
```

Opisane delovne tokove smo združili v en delovni tok z imenom "Migriraj vApp". Shemo delovnega toka smo prikazali na sliki 6.7. Najprej smo uporabili tok "Ugasni vApp". S tem smo izklopili celotno virtualno namensko aplikacijo. Pri prenosu virtualnih strojev je potrebno



Slika 6.7: Delovni tok - migracija virtualne namenske aplikacije.

prenesti vse virtualne stroje znotraj virtualne namenske aplikacije in ustrezno nastaviti števec. Števec vsebuje informacijo o številu virtualnih strojev znotraj virtualne namenske aplikacije, ki čakajo za prenos. Nato za vsak virtualni stroj najprej uporabimo delovni tok "Prenesi OVF" za prenos datotek preko HTTP. Glede na izbrane vhodne parametre virtualnega stroja sestavimo in izvedemo REST operacijo, ki kliče spletno storitev.

6.3.2 Izdelava spletne storitve REST za realizacijo orkestracije

Kot smo že poudarili, obstoječa komponenta za gradnjo hibridnega oblaka - vCloud Connector ne omogoča dostopa do svojih funkcionalnosti preko

programskega vmesnika. Zato smo na strani javnega oblaka postavili spletne storitve REST. Storitve po eni strani omogočajo dostop do potrebnih operacij znotraj javnega oblaka, obenem pa izpostavljajo operacije za dostop preko protokola HTTP. Za namen implementacije zastavljene arhitekture 6.3 smo uporabili vmesnik javnega oblaka vCloud. Uporabili smo zaporedje korakov, katerih rezultat je postavitev virtualne namenske aplikacije.

Za izvedbo migracije smo kreirali metodo, ki sprejme vhodne parametre za nastavitev virtualne namenske aplikacije skupaj z lokacijo datoteke virtualnega stroja. Metodo smo podali v izvorni kodi 6.3.

Izvorna koda 6.3: REST storitev - metoda za migracijo v oblaku.

```
@GET
@Path("migriraj/{potVS}/{imeORG}/{imeVPC}/{imeVNA}")
@Produces("text/plain")
public String migriraj(
        @PathParam("imeVNA") String imeVNA,
        @PathParam("potVS") String potVS,
        @PathParam("imeORG") String imeORG,
        @PathParam("imeVPC") String imeVPC){
...
```

Začetni korak pri vsaki operaciji z vmesnikom vCloud API predstavlja prijava in SSL avtentikacija s certifikatom. Ena izmed možnosti je uvoz in nastavitev SSL certifikatov na strežnik spletne storitve. V našem primeru smo napisali dodatni razred FakeSSLSocketFactory, ki omogoča generiranje certifikata. V produkcijskem okolju je zaželena uporaba digitalno podpisanih certifikatov. Izvorna koda 6.4 prikazuje implementacijo za vzpostavitev povezave z vmesnikom vCloud.

Izvorna koda 6.4: REST storitev - prijava v vCloud Director.

```
String urlString = "https://10.1.1.11/";
VcloudClient.setLogLevel(Level.WARNING);
VcloudClient vcdClient = new VcloudClient(urlString,
```

```
                                        Version.V5_1);
try{
        vcdClient.registerScheme("https", 443,
                FakeSSLSocketFactory.getInstance());
}catch(KeyManagementException e1){
        e1.printStackTrace();
}catch(UnrecoverableKeyException e1){
        e1.printStackTrace();
}catch(NoSuchAlgorithmException e1){
        e1.printStackTrace();
}catch(KeyStoreException e1){
        e1.printStackTrace();
}
```

Preden predlogo postavimo, je potrebno imeti informacijo o organizacijskem virtualnem podatkovnem centru na katerega bomo naložili predlogo. Za ta namen najprej pridobimo referenco organizacije v javnem oblaku, nato pa iz te organizacije izberemo organizacijski virtualni podatkovni center. Implementacijo za izbiro podatkovnega centra smo podali v izvorni kodi 6.5.

Izvorna koda 6.5: REST storitev - izbiranje podatkovnega centra.

```
Organization org = Organization
                    .getOrganizationByReference(vcdClient,
                    vcdClient.getOrgRefByName(imeORG));
Vdc orgVPC = Vdc.getVdcByReference(vcdClient,
                    org.getVdcRefByName(imeVPC));
```

Pri kreiranju nove instance virtualne namenske aplikacije je potrebno nastaviti parametre, s katerimi opišemo naloženo aplikacijo. Najprej smo nastavili osnovne parametre kot so ime in opis virtualne namenske aplikacije, nato smo kreirali objekt predloge. Postopek je prikazan v izvorni kodi 6.6.

Izvorna koda 6.6: REST storitev - kreiranje predloge.

```
UploadVAppTemplateParamsType uploadVNAParametrov =
        new UploadVAppTemplateParamsType();
uploadVNAParametrov.setDescription("Opis VNA");
uploadVNAParametrov.setName(imeVNA);
VappTemplate predloga = orgVPC.createVappTemplate(
                                    uploadVNAParametrov);
```

Predloga, ki smo jo kreirali, mora vsebovati tudi podatke o virtualnih strojih, ki sestavljajo storitve. To nastavimo z uporabo funkcije "uploadOVFFile" za nalaganje OVF datoteke ter "uploadFile" za nalaganje virtualnega diska (datoteka z končnico VMDK). Implementacija za nalaganje datoteke znotraj predloge je podana v izvorni kodi 6.7. Ker operaciji za nalaganje trajata določen čas, smo uporabili princip programskega povpraševanja, tako da smo na določeni interval preverjali stanje OVF deskriptorja oziroma status predloge.

Izvorna koda 6.7: REST storitev - nalaganje datoteke v predlogo.

```
try{
    // nalaganje OVF datoteko
    InputStream OVFFileIs = new BufferedInputStream(
                    new FileInputStream(OVFfile),1024);
    predloga.uploadOVFFile(OVFFileIs, OVFfile.length());
    predloga = VappTemplate.getVappTemplateByReference(
                                vcdClient,
                                predloga.getReference());
    while(!predloga.getResource()
                    .isOvfDescriptorUploaded()){
        try{
            Thread.sleep(5000);
        }catch (InterruptedException e){
            e.printStackTrace();
```

```
        }
        predloga = VappTemplate
                    .getVappTemplateByReference(
                     vcdClient,predloga.getReference());
    }

    // postavitev virtualnega diska
    InputStream vmdkFileIs = new BufferedInputStream(
          new FileInputStream(vmdkFile),1024);
    predloga.uploadFile(imeVNA + "-disk1.vmdk",
                        vmdkFileIs,vmdkFile.length());
    while(predloga.getResource().getStatus() != 8){
        try{
            Thread.sleep(5000);
        }catch(InterruptedException e){
            e.printStackTrace();
        }
        predloga = VappTemplate
                    .getVappTemplateByReference(
                     vcdClient,predloga.getReference());
    }
}catch(FileNotFoundException e1){
    e1.printStackTrace();
}
```

Predlogo virtualne namenske aplikacije, ki smo jo pripravili, je bilo potrebno dodati v organizacijski katalog. V naslednjem delu kode - Izvorna koda 6.8, smo podali proces registracije predloge v katalog storitev. Tukaj izvedemo tri logične operacije: pridobivanje organizacijskega kataloga, nastavitev parametrov ter kreiranje kataloga in dodajanje predloge v katalog.

Izvorna koda 6.8: REST storitev - dodajanje predloge v katalog.

```
Collection<ReferenceType> zbirkaKatalog =
                              org.getCatalogRefs();
Iterator i = zbirkaKatalog.iterator();
ReferenceType izbraniKatalogRef = null;
while (i.hasNext()){
    izbraniKatalogRef = (ReferenceType)i.next();
    if(izbraniKatalogRef.getName()
                            .equals("Katalog1")){
        break;
    }
}
Catalog katalog1 = Catalog.getCatalogByReference(
                               vcdClient,izbraniKatalogRef);
CatalogItemType katalogElementTip =
                    new CatalogItemType();
katalogElementTip.setEntity(predloga.getReference());

XMLGregorianCalendar currentDate = null;
try{
    DatatypeFactory datatypeFactory =
                      DatatypeFactory.newInstance();
    currentDate = datatypeFactory
    .newXMLGregorianCalendar(gregorianCalendar);
}catch(DatatypeConfigurationException e){
    e.printStackTrace();
}
katalogElementTip.setDateCreated(currentDate);
katalogElementTip.setName(imeVNA + "-template");
katalogElementTip.setDescription("Opis");
CatalogItem katalogElement = new CatalogItem(
```

```
                vcdClient, katalogElementTip);
katalog1.addCatalogItem(katalogElement.getResource());
```

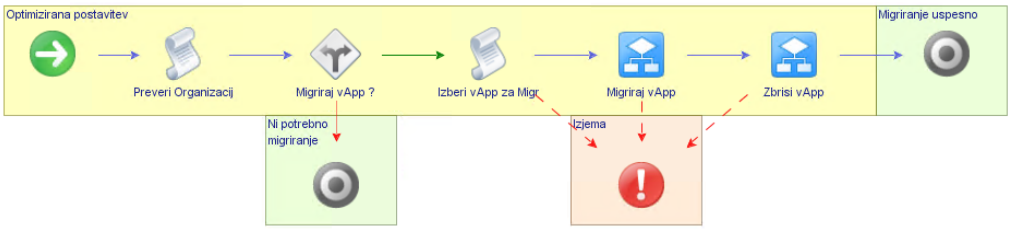
Storitev, prenešeno v javni oblak, je bilo potrebno še instancirati znotraj organizacijskega virtualnega podatkovnega centra. Postavljanje preko vmesnika vCloud poteka tako, da najprej nastavimo parametre predloge, ki se nanašajo na novo virtualno namensko aplikacijo. Po nastaviti parametrov uporabimo metodo instantiateVappTemplate(), ki instancira aplikacijo znotraj organizacije. Primer kode za instanciranje virtualne namenske aplikacije smo podali v izvorni kodi 6.9.

Izvorna koda 6.9: REST storitev - instanciranje virtualne namenske aplikacije iz predloge.

```
InstantiationParamsType parametriInst =
new InstantiationParamsType();

InstantiateVAppTemplateParamsType parametriInstVNA =
new InstantiateVAppTemplateParamsType();

parametriInstVNA.setName(imeVNA);
parametriInstVNA.setSource(predloga);
parametriInstVNA.setInstatiationParams(parametriInst);

Vapp vna = orgVPC.instantiateVappTemplate(
                    parametriInstVNA);
```

6.4 Optimizacija virov v hibridnem oblaku

Optimizacijo virov v hibridnem oblaku lahko uvrstimo med ključne prednosti, ki jih prinese hibridni oblak. To je predvsem zato, ker stanje sistema vzdržujemo na natančno določenem nivoju. Optimizacija v hibridnem oblaku



Slika 6.8: Delovni tok - optimizacija v hibridnem oblaku.

se nanaša predvsem na prilagajanje obremenitev zasebnega oblaka. V primeru, da se preseže določeni nivo zasedenosti virov zasebnega oblaka, se izvede migracija v javni postavitvi hibridnega oblaka. Za izvedbo optimizacije na avtomatizirani način smo izdelali delovni tok podan na sliki 6.8.

Najprej delovni tok preveri obremenitev vseh organizacijskih virtualnih podatkovnih centrov znotraj zasebnega oblaka oziroma organizacije, podobno kot v poglavju 6.2. V primeru, da je izkoriščenost virov znotraj dovoljene vrednosti, se delovni tok konča, v obratnem primeru, pa tok nadaljuje z migracijo ene izmed storitev. V našem primeru izberemo prvo izvajajočo se storitev znotraj izbranega virtualnega podatkovnega centra. Storitev nastavimo kot vhodni parameter delovnega toka za migracijo ter izvedemo delovni tok. V primeru uspešne migracije uporabimo delovni tok “Zbrisi vApp”, ki izbriše virtualne namenske aplikacije in s tem sprosti zasedene vire v zasebnem oblaku.

Končni korak pri optimizaciji virov je nastavitev samodejnega izvajanja delovnega toka. Za ta namen smo ustvarili opravilo, ki v določenih intervalih zaganja delovni tok za optimizacijo delovanja zasebnega oblaka.

Poglavje 7

Zaključek

Cilji diplomskega dela so bili opis osnovnih konceptov orkestracije in avtomatizacije v oblaku; izvedba primerjalne analize obstoječih rešitev za orkestracijo; opis osnovne arhitekture orodja za orkestracijo vCenter Orchestrator; ter izdelava dveh praktičnih primerov, ki obravnavajo tako zasebni kot tudi hibridni oblak.

V diplomskem delu smo obravnavali avtomatizacijo in predvsem orkestracijo v oblaku, ki postajata vse bolj pomembni funkcionalnosti računalništva v oblaku. Ugotovili smo, da praktično vsi največji ponudniki platform za postavitev oblaka vključujejo tudi komponente za izvedbo orkestracije. Tem komponentam je skupno, da omogočajo visoko stopnjo integracije s komponentami lastnega oblaka, podpora za zunanje platforme pa je minimalna.

Platforma VMware, kot ena izmed najbolj uporabljenih platform za virtualizacijo glede na analizi Gartner [33] ali Forrester [34], ponuja velik nabor komponent za upravljanje oblaka, od infrastrukture do nivojev storitve. Komponenta za orkestracijo vCenter Orchestrator, ki smo jo uporabili pri gradnji delovnih tokov, nam je olajšala delo zaradi velikega števila pripravljenih predlog delovnih tokov.

Ugotovili smo, da vgrajeni vtičniki za delo s komponentami vCloud in vCenter pokrivajo dober delež funkcionalnosti, ki so na voljo v posameznih komponentah običajno preko čarovnikov. Posledično, smo z uporabo ob-

stoječih delovnih tokov in aktivnosti ter s kreiranjem novih delovnih tokov uspešno izvedli celoten postopek postavitve storitve Apache Tomcat.

Pri analizi zmožnosti uporabe orodja za orkestracijo smo ugotovili, da ni vgrajenih gradnikov za integracijo s hibridnim oblakom. Čeprav VMware ponuja ločeno komponento za postavitev hibridnega oblaka - vCloud Connector, je le-ta zelo omejena glede funkcionalnosti in ne ponuja možnosti programske uporabe obstoječih funkcij. V diplomskem delu smo prikazali kako izvesti nujne operacije za komunikacijo med zasebnim in javnim oblakom, ter prenos delovnih obremenitev (storitev) iz enega v drugega. Pri tem smo zastavili arhitekturo, ki se v določeni meri zgleduje po arhitekturi vCloud Connector s tem, da omogoča izpostavljanje funkcionalnosti hibridnega oblaka preko REST vmesnika. Arhitektura predstavlja samo praktični primer in jo je mogoče dodatno optimizirati ter izboljšati.

Literatura

[1] A. T. Velte, T. J. Velte, R. Elsenpeter "Cloud Computing: A Practical Approach", 2010, str. 30

[2] (2012) Amazon Elastic Compute User Guide, Dostopno na: `http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf`

[3] (2006) Amazon Simple Storage Service Getting Started guide. Dostopno na: `http://s3.amazonaws.com/awsdocs/S3/latest/s3-gsg.pdf`

[4] (2013) Amazon Relational Database Service Getting Started Guide. Dostopno na: `http://awsdocs.s3.amazonaws.com/RDS/latest/rds-gsg.pdf`

[5] (2012) Amazon CloudFront Developer Guide. Dostopno na: `http://s3.amazonaws.com/awsdocs/CF/latest/cf_dg.pdf`

[6] (2011) VMware vCenter Converter User's Guide, vCenter Converter Standalone 5.0. Dostopno na: `https://www.vmware.com/pdf/convsa_50_guide.pdf`

[7] B. Sosinsky, "Cloud Computing Bible", 2011, str. 134

[8] P. Mell, T. Grance, The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technology, September 2011.

[9] (2012) Which Open Source Cloud Platform Should You Use. Dostopno na: http://midsizeinsider.com/en-us/article/which-open-source-cloud-platform-should

[10] (2012) Comparing Open Source Private Cloud (IaaS) Platforms. Dostopno na: http://cdn.oreillystatic.com/en/assets/1/event/80/Comparing%20Open%20Source%20Private%20Cloud%20Platforms%20Presentation.pdf

[11] (2012) 10 most powerful IaaS companies. Dostopno na: http://www.networkworld.com/supp/2012/enterprise2/040912-ecs-iaas-companies-257611.html?page=1

[12] OpenStack software documentation. Dostopno na: https://wiki.openstack.org/wiki/Documentation#OpenStack_software_documentation

[13] (2013) Eucalyptus 3.2.1 Administration Guide. Dostopno na: http://www.eucalyptus.com/docs/3.2/admin-guide-3.2.1.pdf

[14] (2012) Cloud Infrastructure Configuration Management Tools and Options. Dostopno na: http://communities.intel.com/community/datastack/blog/2012/06/27/cloud-infrastructure-configuration-management

[15] (2011) Top Tools For Opensource Cloud Computing. Dostopno na: http://www.toolsjournal.com/cloud-articles/item/221-top-tools-for-opensource-cloud-computing

[16] (2012) Puppet Documentation. Dostopno na: http://downloads.puppetlabs.com/docs/puppetmanual.pdf

[17] (2012) Opscode Chef Wiki. Dostopno na: http://wiki.opscode.com/download/attachments/1179727/chef-170212-2137-12.pdf?version=1&modificationDate=1329515989000

[18] (2012) Gartner releases its Evaluation Criteria for Cloud Management Platforms. Dostopno na: http://cloudcomputing.info/en/news/2012/07/gartner-releases-its-evaluation-criteria-for-cloud-management-platforms.html

[19] (2011) Forrester Evaluates Private Cloud Vendors - Platform Top Media Scoring. Dostopno na: http://platformcomputing.blogspot.com/2011/05/forrester-scores-private-cloud-vendors.html

[20] (2012) 30 Private Cloud Computing Vendors to Consider. Dostopno na: http://www.webopedia.com/DidYouKnow/private-cloud-computing-vendors-to-consider.html

[21] (2012) Cisco Process Orchestrator. Dostopno na: http://www.cisco.com/en/US/prod/collateral/netmgtsw/ps6505/ps11038/ps11100/1110E0910_cte_orchestrator.pdf

[22] (2011) System Center Orchestrator 2012 Getting Started Guide. Dostopno na: http://www.technicalwritingprofessional.com/images/Orch2012_Get_Started.pdf

[23] (2012) Automation Repetitive Tasks with BMC Atrium Orchestrator. Dostopno na: http://documents.bmc.com/products/documents/16/06/241606/241606.pdf

[24] (2012) AWS Flow Framework Developer Guide(Java). Dostopno na: http://awsdocs.s3.amazonaws.com/swf/latest/swf-aflow.pdf

[25] (2012) VMware vCloud Automation Center. Dostopno na: http://www.vmware.com/files/pdf/vcloud/vmware-vcloud-automation-center-datasheet.pdf

[26] (2013) Installing and Configuring VMware vCenter Orchestrator. Dostopno na: http://pubs.vmware.com/vsphere-51/topic/com.vmware.ICbase/PDF/vcenter-orchestrator-51-install-config-guide.pdf

[27] (2013) Active Directory Architecture. Dostopno na: http://technet.microsoft.com/en-us/library/bb727030.aspx

[28] (2009) Rhino overview. Dostopno na: https://developer.mozilla.org/en-US/docs/Rhino/Overview

[29] (2012) vCloud Director User's Guide. Dostopno na: http://pubs.vmware.com/vcd-51/topic/com.vmware.ICbase/PDF/vcd_51_users_guide.pdf

[30] (2012) VMware vCenter Operations Manager Getting Started Guide. Dostopno na: http://www.vmware.com/pdf/vcops-56-getting-started-guide.pdf

[31] (2013) Developing Plug-Ins with VMware vCenter Orchestrator. Dostopno na: http://communities.vmware.com/servlet/JiveServlet/previewBody/20394-102-3-28663/vcenter-orchestrator-51-develop-plugins-guide.pdf

[32] (2012) Installing And Configuring vCloud Connector. Dostopno na: http://pubs.vmware.com/hybridcloud-20/topic/com.vmware.ICbase/PDF/vCloudConnector_20_InstallConfigure.pdf

[33] (2012) Magic Quadrant for x86 Server Virtualization Infrastructure. Dostopno na: http://www.gartner.com/technology/reprints.do?id=1-1B2IRYF&ct=120626&st=sg

[34] (2013) 2013 Server virtualization predictions: driving value above and beyond the hypervisor. Dostopno na: http://blogs.forrester.com/dave_bartoletti/13-02-01-2013_server_virtualization_predictions_driving_value_above_and_beyond_the_hypervisor