ELEKTROTEHNIŠKI VESTNIK 89(4): 143–149, 2022
PREGLEDNI ZNANSTVENI ČLANEK

Avtomatizacija in orkestracija v sodobnem razvoju programske opreme

Anže Luzar in Mojca Ciglarič

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana, Slovenija
E-pošta: mojca.ciglaric@fri.uni-lj.si

Povzetek. Okrajšava DevOps predstavlja povezavo med dvema področjema dela v IT, namreč med razvojem in operativo, novejša izpeljanka DevSecOps jima pridružuje še področje varnosti. Med dobre prakse področja sodita avtomatizacija in orkestracija, vendar se ta dva pojma pogosto uporabljata nekonsistentno. Članek predstavi in določi osnovne pojme in principe, ki so ključni za razumevanje in za strukturiranje tega področja. Kot avtomatizacijo razumemo predvsem uporabo orodij za posamezne naloge, kot so nameščanje in integracija, orkestracija pa predstavlja kompleksnejši postopek razvrščanja večjih skupin nalog, pogosto v povezavi z oblakom. Orodja in standardi se še razvijajo, združljivost je omejena, medtem ko so prednosti pravilne uporabe lahko zelo velike. Članek pregledno predstavi področje, vlogo pri razvoju aplikacij, pomanjkljivosti tehnologij in orodij ter glavne izzive.

Ključne besede: avtomatizacija, DevOps, oblak, orkestracija

Automation and orchestration in modern software development

The acronym DevOps represents the interdependence between two IT areas, development and operations, while DevSecOps includes also security area. Although automation and orchestration represent well known approaches, the terms are often used inconsistently. This paper presents and defines the main concepts and principles needed for understanding and structuring the area. While the term automation simply means the use of tools for performing individual tasks, such as installation and integration, orchestration is a more complex process of synchronising and managing larger groups of tasks, often running in the cloud. Tools and standards are still being developed and their compatibility is limited, however the benefits of its proper use can be substantial. The paper overviews the field, the role in application development, the shortcomings of technologies and tools, and the main challenges.

Keywords: automation, DevOps, cloud, orchestration

1 UVOD

Avtomatizacija na področju IT prestavlja uporabo informacijskih orodij za samodejno izvajanje naloge, kot je na primer nameščanje, brez človeških intervencij. Orkestracija pa predstavlja kompleksnejši postopek obvladovanja večjih skupin avtomatiziranih nalog, pogosto v povezavi z oblakom, ki se usklajeno izvajajo kot informacijski proces ali delovni tok [22]. Tako v manjših podjetjih kot tudi v velikih korporacijah lahko uporaba orodij za avtomatizacijo in orkestracijo poleg tehnoloških prednosti predstavlja tudi ključno poslovno prednost in bistveno zmanjšanje operativnih stroškov (OpEx) [3]. DevOps kot pristop k razvoju informacijskih rešitev predstavlja povezovanje nekoč ločenih skupin strokovnjakov za razvoj in operativno delovanje informacijskih rešitev z namenom zagotoviti bolj učinkovit, hiter in varen razvoj in dostavo rešitev. Avtomatizacija in orkestracija se v tem okviru intenzivno uporabljata. Poleg pričakovanih pozitivnih učinkov in prihrankov pa se kažejo tudi nekateri pomisleki.

Omenjena orodja se uporabljajo tudi za selitev in razmestitev informacijskih rešitev v oblaku. Razvijalci si želijo neodvisnosti od oblačne platforme, ponudniki storitev v oblaku pa zagotavljajo predvsem združljivost v okviru lastnih storitev. To je za uporabnike lahko omejujoče, na primer pri menjavi ponudnika ali migraciji dela svojih storitev v drug oblak. Med večja tveganja pri oblačni migraciji sodita združljivost in nevarnost zaklepanja na določenega ponudnika [4]. Zato je pomembna tudi združljivost razmestitvenih orodij, ki rešujejo različne težave in pogosto ne podpirajo orkestracije na različne ponudnike. Z izbiro orkestratorja in orodij za orkestracijo so izbrane tudi prednosti in slabosti njihovega procesa orkestracije. Trenutno ni opaziti značilnih presekov med podporo orkestracije in avtomatizacije in oblačnih storitev. V tem članku želimo pregledno pojasniti pristop DevOps in z njim povezana pojma avtomatizacije in orkestracije, njihovo vlogo v sodobnih procesih na področju IT in pogoste izzive, ki se pojavljajo v povezavi z njimi. Z jasnejšim pregledom in boljšim razumevanjem področja bodo namreč lažje tudi odločitve za izbiro primernih orodij in tehnologij.

Prejet 7. marec, 2022
Odobren 3. maj, 2022

2 PRISTOP DEVOPS

Izraz DevOps izvira iz angleške fraze Development and Operations, kar bi lahko prevedli kot razvoj in operativno delovanje informacijskih rešitev. Pristop DevOps v ospredje postavi komunikacijo in sodelovanje med razvijalci informacijskih rešitev in tistimi strokovnjaki s področja IT, ki so zadolženi za operativno delovanje teh rešitev, in se uporablja predvsem pri avtomatiziranem uvajanju programskih sprememb in sprememb na področju informacijske infrastrukture [1].

Bolj kot za neke vnaprej predpisane postopke gre za delovno kulturo, ki omogoči boljše komuniciranje med različnimi področji IT v podjetju. Pristop DevOps namreč poveže tiste zaposlene, med katerimi se v primeru pomanjkljive komunikacije lahko zgodi prelaganje odgovornosti za morebitne težave ali napake, kar je negativno, vodi v podaljšanje časa razvoja [1] in upadanje kakovosti.

Izraz DevOps se je pojavil leta 2009 kot nov pristop k razvoju informacijskih rešitev. Glavni cilj novega pristopa je bil, da bi z njegovo uporabo organizacije lahko hitreje ustvarjale in posodabljale svoje informacijske rešitve in storitve [2]. V začetku je bilo gibanje DevOps zanimivo predvsem za mlajša in zagonska podjetja, kasneje pa se je razširilo na vse tipe organizacij in na vse konce sveta. Zamisel o tem pristopu je dobil Patrick Debois, ki je bil eden izmed največjih navdušencev agilnih metodologij. Model DevOps torej izhaja iz krogov podpornikov agilnih metodologij, katerih cilj je usklajenost ekipe razvijalcev informacijskih rešitev in operativnih skupin z namenom, da se izboljšajo izdelki v vseh fazah življenjskega cikla programske opreme (razvoj, testiranje, uvajanje, delovanje). Cilj pristopa DevOps je doseganje določenih poslovnih ciljev, kot so hiter in zanesljiv razvoj, zniževanje stroškov in tveganja in zadovoljstvo naročnikov, kar za podjetja lahko prestavlja pomembno konkurenčno prednost [2]. DevOps je povezan tudi s poslovnimi procesi, saj si prizadeva za njihovo optimizacijo, pri čemer poudarja, da se to lahko doseže z uporabo procesov CI/CD - neprekinjene integracije in neprekinjenega nameščanja [6].

DevOps predstavlja združevanje dveh do tedaj ločenih taborov: razvijalcev in ekip za podporo operativnemu delovanju informacijskih tehnologij. Prve vodi želja po spremembah, drugi pa si prizadevajo za stabilnost produktov, ki jih ponudijo strankam, saj si te želijo varnosti in zanesljivosti. Vsaka nova funkcionalnost, ki jo razvijalec doda, lahko ogrozi cilje operativne ekipe in lahko zahteva prilagajanje infrastrukture [8].

DevOps uporablja orodja, namenjena predvsem podjetjem s področja informacijske tehnologije. Ta filozofija poudarja avtomatizacijo, standardizacijo in aktivnejši odnos vseh vpletenih. V literaturi se za pristop DevOps navajajo značilne vidne prednosti in pomanjkljivosti.

Glavne pozitivne strani tega pristopa so predvsem [5]:

- agilnost razvoja, usklajenost ekip, obvladovanje tveganj
- hitrejši razvoj bolj kakovostnih in varnejših rešitev
- neprestano izboljševanje procesa

Najpogostejši negativni vidiki so:

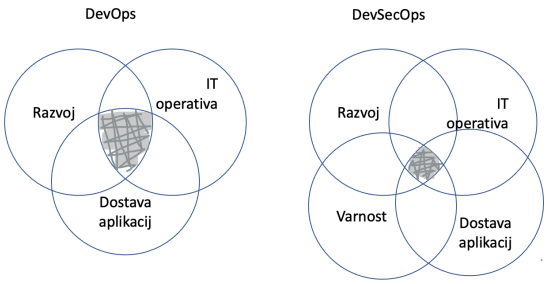
- morebitno opuščanje predhodnih uspešnih praks, saj mnogi menijo, da so te zastarele; DevOps pristop morda ne bo rešil težav.
- potrebna je sprememba organizacijske kulture, za nekatere skupine ta pristop morda sploh ni ustrezen.

Ob vsem zahtevnem razvoju in pritisku na hitro vzpostavitev storitev ne smemo pozabiti na varnost (angl. security). Da se izkoristi agilnost in vse prednosti prakse DevOps ter razvije celovito rešitev, je potrebno, da ima slednja tudi integrirane ustrezne varnostne mehanizme. Če v pristop DevOps vključimo še stalno komunikacijo z ekipo, ki skrbi za varnost, dobimo nov sestavljen pojem DevSecOps, kar je okrajšava za DEVelopment-and-SECurity-OPerationS [9].

V klasičnem slapovnem razvoju je bila varnost naloga ločene ekipe, ki je za dodajanje varnostnih mehanizmov in kontrol v izdelek poskrbela ob zaključku razvoja aplikacije [9]. Nekdaj to ni predstavljalo težav, saj so razvojni cikli trajali tudi po več mesecev ali let. Danes so v modernem razvoju ti cikli veliko krajši in jih merimo v dnevih ali tednih, kar je tudi ena od zahtev DevOps prakse, a rezultati brez zagotovljenega ustreznega nivoja varnosti razočarajo. V kolikor je zaradi pomanjkljivih varnostnih kontrol prišlo do vdorov in napadov, je breme krivde padlo na operativno ekipo in sistemske inženirje, saj so bili razvijalci po večini nedotakljivi, ko so opravili svojo glavno nalogo, namreč da programska koda dela tisto, za kar je bila načrtovana. V izogib takšnih težav se izkaže, da je potrebno varnostne mehanizme in preverjanje programske opreme načrtovati in izvajati že v zgodnjih fazah znotraj razvojnega cikla, saj dodatek zgolj avtentikacije in šifriranja na koncu razvoja ni bil več dovolj za zagotovitev vedno višjih zahtev po varnosti.

Kultura DevOps je tako z vključitvijo varnostne ekipe k razvoju doživela metamorfozo in se preoblikovala v DevSecOps, ki ima vgrajeno tudi varnost in poudari to, da se že od samega začetka infrastrukturo informacijske rešitve načrtuje tudi z varnostnega vidika. Posledično razvijalci med programiranjem neprestano upoštevajo tudi varnost in uporabljajo dobre prakse razvoja varnostnih mehanizmov ter svoje rešitve delijo z varnostno ekipo, ki jih usmerja in jim svetuje, katere grožnje bi še lahko predstavljale tveganje za zasnovani sistem. Včasih je za dobro sodelovanje med ekipama potrebno dodatno varnostno izobraževanje za razvijalce, da ti res ponotranjijo upoštevanje varnost [10].

Z varnostnimi testiranji znotraj razvojnih ciklov bi lahko prišlo do zakasnitev, zato je pomembno, da so tudi varnostna testiranja kar se da avtomatizirana in da



Slika 1: Primerjava pristopov DevOps (v preseku razvoja, operative in dostave aplikacij) in DevSecOps (v preseku razvoja, operative, dostave aplikacij in varnosti).

ne preprečujejo ali upočasnjujejo razvoja. Avtomatiziran pristop DevSecOps se danes pogosto dosega ob pomoči orodij neprekinjene integracije in namestitve (orodja CI/CD) ter z implementacijo v obliki mikrostoritev, zapakiranih v kontejnerje [11].

3 AVTOMATIZACIJA

Avtomatizacija je tehnološki princip, kjer za proces ali proceduro, ki se izvaja, ni potrebna človeška prisotnost oz. je človek prisoten le minimalno [12]. Sam pojem izvira s področja industrije še iz časa Henryja Forda, ko so v tovarni Ford naredili prvi tekoči trak. Takrat se je avtomatizacija nanašala predvsem na uporabo strojev, ki so opravljali ponavljajoče se naloge [13].

Pri avtomatizaciji v računalništvu gre predvsem za uporabo orodij za avtomatizacijo pri procesih in nalogah, ki so povezane z nameščanjem aplikacij in integracijo, a v ospredje prihaja tudi avtomatizacija v oblačnih storitvah. Kadar uporabljamo pojem avtomatizacija, mislimo na majhno enoto ali na eno samo nalogo, ki se jo avtomatizira, za razliko od orkestracije, ki se nanaša na razvrščanje skupine nalog [14].

Avtomatizacija lahko zajema različne tehnologije, arhitekturne oblike, kot so kontejnerji, in se uporablja v različnih pristopih, kot je tudi DevOps. Pogosto jo srečamo v računalništvu v oblaku, pri uporabi v robnih napravah, v varnostnih mehanizmih, pri testiranju in nadzoru delovanja aplikacij [7].

Med glavna področja, kjer se danes uporablja avtomatizacija na področju IT, sodijo [26] zagotavljanje infrastrukture, upravljanje konfiguracij, nadzor in testiranje, nameščanje, gradnja in dostava aplikacij ter varnost in doseganje skladnosti.

Sodobna informacijska okolja so zahtevna in jih sestavljajo mnoge kompleksne komponente, tako strojne kot programske. Avtomatizacija procesov, upravljanja, nadzora in vzdrževanja omogoča, da organizacije obdržijo nadzor nad svojo infrastrukturo in stroški, hkrati z avtomatizacijo prihranijo čas in se lahko osredotočijo na zagotavljanje novih funkcionalnosti za svoje uporabnike. Vsaka avtomatizacija se mora začeti s ciljem, ki določa, kaj sploh bomo avtomatizirali. Pomemben pogoj za uspešno avtomatizacijo je standardizacija poslovnih procesov, saj lahko le z dobro definiranimi procesi brez napak v celoti izkoristimo prednosti avtomatiziranega informacijskega okolja. Ena izmed glavnih prednosti, ki jih ponuja avtomatizacija podjetjem in razvijalcem, je višja kakovost storitev, saj avtomatizacija omogoči opraviti naloge hitreje in bolje tudi v primeru, ko se zahteve pogosto spreminjajo. Druga prednost je prihranek časa in virov, saj se lahko razvojna skupina usmeri k razvijanju novih storitev, ki dejansko prinesejo dobiček. Z uporabo avtomatizacije se lahko nadejamo večje varnosti in zakonske skladnosti, saj lahko zagotovimo doslednost izvajanja varnostne politike. Kljub avtomatizaciji se sicer ne moremo povsem izogniti napakam, a standardizirani in avtomatizirani procesi vseeno zagotavljajo nižje tveganje in manj skrbi [15].

Kljub svojim pozitivnim stranem avtomatizacija prinaša tudi določena tveganja in pomanjkljivosti. Avtomatizacija zahteva višjo kompleksnost infrastrukture in okolja, potrebna je tudi investicija v orodja in izobraževanje. Posledica so višji stroški, kar je v nasprotju s splošnim prepričanjem, da avtomatizacija organizacijam prinaša le prihranek. Za uspešno in učinkovito avtomatizacijo je potrebno poglobljeno poznavanje avtomatizacijskih orodij in tehnologij, kar lahko pomeni potrebo po novih zaposlitvah, ali časovno zahtevno izobraževanje za obstoječe strokovnjake. Avtomatizacija potegne za sabo tudi spremembe v implementaciji procesov in potrebo po stalnem nadzoru in testiranju, da se zagotavlja pravilno delovanje in doseganje ciljev razvoja [16].

Za velika podjetja in organizacije je avtomatizacija je ključnega pomena, saj predstavlja vzvod, ki lahko izboljša operativne procese in omogoča vpeljavo koristnih orodij. Avtomatizacija skupaj z orkestracijo predstavlja nov pristop - avtomatizacijo na ravni upravljanja infrastrukture, kar pravzaprav pomeni transformacijo storitev v organizaciji in poenostavitev njihovega zagotavljanja. Na področju informacijske infrastrukture se namreč izvaja kopica ponovljivih nalog, ki lahko po nepotrebnem porabljajo čas in so obenem tudi izhodišče za človeške napake [18].

4 ORKESTRACIJA

Orkestracija ali razmestitev je avtomatizirano konfiguriranje, ki skrbi za nadzor in koordinacijo računalniških sistemov, informacijskih rešitev in storitev ter pripomore k lažji izvedbi kompleksnejših nalog in skupin nalog. Orkestriranje oziroma razmeščanje z uporabo orkestracijskih orodij ali orkestratorjev rešuje težavo povezovanja večjega števila avtomatiziranih nalog in njihovih konfiguracij na različnih sistemih [20].

Orkestracija predstavlja razporejanje in integracijo nalog. Osredotoča se na to, kako povezati procese ne glede na tip, pri čemer ločuje med združevanjem

procesov tehnične narave in poslovnimi procesi [14]. Velikokrat orkestracijo srečamo v povezavi z virtualizacijo omrežnih funkcij, kjer govorimo o upravljanju in orkestraciji (angl. Management and Orchestration - MANO) [19].

S pomočjo orkestracije lahko vsak proces predstavimo z delovnim tokom in ga tako zapakiranega večkrat uporabimo, prav tako lahko nato povezujemo več delovnih tokov oz. podprocesov med sabo v večji proces in s tem orkestriramo tudi večje sisteme, na primer celotne oblačne sisteme [17].

Uporaba orkestracije lahko za podjetja in razvijalce prestavlja mnoge prednosti, saj pomaga pri poenostavitvi in optimizaciji ponavljajočih se procesov, s tem dodatno podpira celoten proces DevOps in podpira hitrejše nameščanje aplikacij. Ureditev časovno potratnih nalog z orkestracijo vpliva tudi na prihranek časa in zmanjšanje stroškov ter omogoča boljšo produktivnost razvojnih ekip, zagotavlja tudi predvidljivost izvedbe ponovljivih avtomatiziranih procesov in s tem zmanjša prostor za (človeške) napake ter ustvari zanesljivo informacijsko okolje [20].

Seveda orkestracija prinaša tudi določene izzive in pomanjkljivosti. Kljub olajšanju skrbi, povezanih z razporejanjem nalog je še vedno potrebno narediti dober načrt za implementacijo. Uporaba orkestracije sicer zmanjša možnost za napake, vendar jih ne more preprečiti. Že manjša napaka v procesu orkestracije ima lahko velike posledice. Vzpostavitev orkestracije je za razvojno ekipo izziv, saj se mora ta prilagoditi in ponotranjiti nove procese, osvojiti uporabo različnih orkestratorjev oz. razmestitvenih orodij in med njimi izbrati primerno glede na zastavljen namen in cilje [27].

Orkestracija predstavlja način, kako definiramo delovni tok, ki je sestavljen iz logičnega zaporedja preprostih (avtomatiziranih) nalog. Te naloge vodijo k zastavljenim ciljem, doseganje katerih je tudi namen orkestracije. Glede na področja uporabe ločimo več vrst orkestracije, najbolj tipične so naslednje tri:

- orkestracija v oblaku (angl. Cloud orchestration);
- orkestracija storitev (angl. Service orchestration) in
- orkestracija izdaj (angl. Release orchestration).

Orkestracijo v oblačnih storitvah se običajno uporablja za vzpostavitev virtualnih naprav, za shranjevanje podatkov, za konfiguracijo omrežja, za postavitev aplikacij in njihovih funkcij, vsak ponudnik pa nudi še lastne specifične storitve. Sodoben pristop z orkestracijo olajša delo sistemskim inženirjem, ki so morali pred tem vse navedene naloge opravljati najprej ročno, kasneje pa s pomočjo orodij za avtomatizacijo. Ta se sedaj uporabljajo za implementacijo nalog, ki se kasneje orkestrirajo.

Storitvena orkestracija naj bi vzpostavljala celotno rešitev za dostavo popolne storitve, kar pomeni, da bi morala podpreti vse korake od načrtovanja informacijske rešitve do njene vzpostavitve v produkcijskem okolju. V resnici je pogosteje tako, da storitve uporabljajo orodja, ki nudijo različne zmogljivosti v obliki funkcij znotraj aplikacije tako, da ni potrebno zajeti vseh prvotno definiranih delov storitve. Pri storitvah se orkestracija bolj kot na izvajanje osnovnih nalog osredotoča na sledenje stanja teh nalog, zato včasih izvajanje nekaterih nalog sploh ni vključeno.

Orkestracija izdaj uporablja tako imenovana ARO orodja (angl. Application Release Orchestration Tools, ARO), s katerimi dosežemo kombinacijo avtomatizacije za namestitev aplikacije, cevovodov, upravljanje okolja aplikacije in možnost hitrega kreiranja izdaj, s katerimi postopoma z verzioniranjem izboljšujemo kakovost naše aplikacije. Ta orodja omogočajo, da izdajamo različice aplikacije glede na svoj način razvoja, saj upoštevajo raznolikost ekip po pristopu DevOps in tudi različne metodologije razvoja programske opreme, kot npr. agilni razvoj po metodologiji Scrum [28].

Razlika med avtomatizacijo in orkestracijo je pomembna, čeprav se v teoriji in v praksi procesa včasih zamenjujeta ali meja med njima ni jasna. Izvajanje posameznih nalog, ki so zadolžene za neko operacijo, spada pod avtomatizacijo, medtem ko gre pri orkestraciji za združevanje in upravljanje množice avtomatiziranih nalog, kar omogoči, da se proces izvede v celoti. Avtomatizacija pomaga narediti naloge bolj učinkovite in zmanjšati prisotnost človeka, orkestracija pa poveže avtomatizirane procese tako, da se ti lahko izvajajo samodejno in v logičnem zaporedju [21].

5 RAZVOJ INFORMACIJSKIH REŠITEV

Avtomatizacija in orkestracija pomembno vplivata na razvoj informacijskih rešitev. Nekoč so se za razvoj rešitev uporabljali drugačni pristopi in orodja kot danes. Infrastrukture niso vključevale novejših tehnologij in arhitektur, kot sta na primer oblak ali kontejnerizacija. Značilna arhitektura aplikacije je bila nekoč monolitna, združevala je vse komponente aplikacije na enem mestu, tipično je bila močno odvisna od platforme in infrastrukture in ni uporabljala abstrakcije po komponentah, ki bi omogočala, da aplikacija lahko teče povsod. Življenjski cikel takih aplikacij je vključeval namestitev aplikacije na lokalnih strežnikih, kar je bilo povezano tudi s počasno dostavo in manj učinkovitim testiranjem. Poudarek na varnostnih mehanizmih je bil bistveno manjši, komunikacija med skupinami ni tekla po principih DevOps, neprekinjena integracija in dostava še nista bila uveljavljena pojma [29].

Informacijske rešitve postajajo vse bolj zmogljive in kompleksne [25]. Tradicionalni življenjski cikel je začel postajati neobvladljiv, zato se je spremenila tudi dostava teh rešitev do uporabnikov in oblikovali so se principi razvoja, ki zagotavljajo hitro in učinkovito izgradnjo robustnih in kompleksnih aplikacij, ki jih je možno hitro in varno dostaviti v uporabo in jih kasneje po potrebi širiti. Osnovna vodila lahko povzamemo v treh točkah,

in sicer [23]:

- aplikacija naj bo razdeljena na manjše dele;
- aplikacija naj bo načrtovana za lažji razvoj in vzdrževanje;
- aplikacija naj bo povezljiva.

Prvo vodilo svetuje, da ne razvijamo velikih monolitnih aplikacij, ampak aplikacijo razdelimo na manjše samostojne dele, kar zmanjša pritisk na razvijalce, omogoči lažje in hitrejše testiranje in tudi lažje spreminjanje ter popravke pri novih verzijah. Drugo vodilo spodbuja tak sistem razvoja in dostave aplikacije, ki omogoča razvijalcem lažje delo, saj postaneta koda in arhitektura aplikacije razumljivejši, orodja DevOps pa zagotavljajo tudi učinkovito dostavo. Bistvo tretjega vodila je, da komunikacija med deli aplikacije poteka preko omrežja in ne preko notranjega medpomnilnika. To ustreza načinu dela porazdeljenih razvojnih ekip, pospeši namestitev aplikacije, zagotavlja robustnost in prilagodljivost na različna okolja.

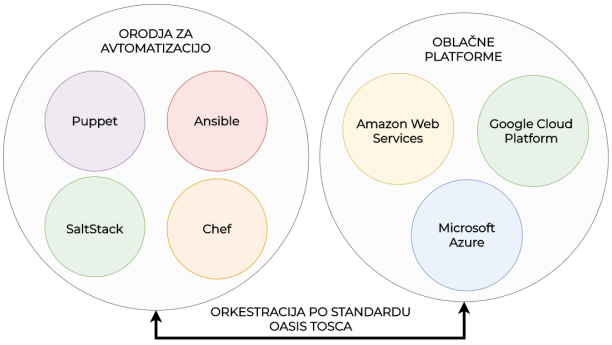
Vsa tri vodila ustrezajo sodobnemu načinu razvoja informacijskih rešitev, pristopu DevOps, podpirajo neprekinjeno dostavo aplikacij, uporabo kontejnerjev (npr. Docker, LXC, Vagrant) in orkestracijskih ogrodij zanje (npr. Kubernetes), uporabo mikrostoritev in s tem povezane dobre prakse. Sodobne aplikacije poleg vsega naštetega tipično podpirajo več različnih odjemalcev in za dostop do podatkov uporabljajo aplikacijski programski vmesnik (angl. Application Program Interface, API) s protokolom HTTPS. Podatki so na voljo v berljivem formatu, kot je npr. JSON, za samo implementacijo aplikacije se uporablja sodoben sklad s priljubljenimi programskimi jeziki, kot so Java, Python, Node, Ruby, PHP, Go itn., kar razvijalcem nudi lažji razvoj rešitev [30].

Za sodoben razvoj informacijskih rešitev je značilen tudi premik namestitve z lokalnih strežnikov v oblačne infrastrukture in uporaba oblačne orkestracije. Rešitev tako gostuje v oblaku oziroma na enem ali več strežnikih v podatkovnem centru ponudnika oblačne storitve. Za pravilno načrtovane rešitve v oblaku je značilno, da so skalabilne, neprekinjeno dostopne in podpirajo virtualne naprave. Primeri teh virualizacijskih oblačnih tehnologij so na primer VMware Cloud foundry, Google apps Engine, Microsoft Azure, Appcara, Salesforce (Heroku and Force.com), AppFog, Engine Yard, Standing Cloud, Mendix in še mnogi drugi [31].

6 IZZIVI NA PODROČJU DEVOPS

Pristop DevOps skupaj z avtomatizacijo in orkestracijo prinaša številne prednosti. Storitve, aplikacije in funkcije, ki so prej delovale na samostojnih strežnikih, se selijo v oblak, kar razvijalce razbremeni opravil, povezanih s postavitvijo primerne infrastrukture in zagotavljanjem zanesljivosti lastnih strežnikov. Odgovornost za kakovost storitve se tako v večjem delu prenese na ponudnika oblačne storitve.

Ob predpostavki, da bo informacijska rešitev tekla na oblačni infrastrukturi, konfiguracijo in celotni proces namestitve obvladujeta orkestracija in avtomatizacija. Zdi se, da sta ta procesa z razmahom oblaka postala izjemno močno povezana s postavitvijo, saj lahko zakrijeta specifike oblačne infrastrukture in razvijalcem, ki z orkestratorjem poganjajo skripte z avtomatiziranimi nalogami, za samo namestitev ni potrebno v globino poznati oblačne storitve, ampak se lahko za pravilno interpretiranje in uporabo oblačnih funkcij zanesejo na orkestrator.



Slika 2: Vrzel med orodji in platformami oblačnih storitev bi lahko premostili z neodvisnim standardom, kot je na primer Oasis Tosca.

V trenutnih razmerah lahko za vzpostavitev infrastrukture, na kateri bo tekla informacijska rešitev, izbiramo med številnimi različnimi oblačnimi platformami, med katerimi izstopajo Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, Alibaba Cloud, IBM Cloud, VmWare Cloud, Oracle Cloud in drugi. Vse naštete platforme se uporabljajo z istim namenom, a vsaka od njih ima svoje značilnosti in posebnosti. V primeru potrebe po namestitvi neke aplikacije ali njenih posameznih delov na različne platforme pa lahko naletimo na težave. Ne poznamo namreč primernega enotnega načina za učinkovito dostavo aplikacij na različne ponudnike z upoštevanjem relacij med posameznimi deli aplikacije. Med orodji za orkestracijo oz. avtomatizacijo in ponudniki oblačnih storitev je namreč vrzel, kar vizualno prikazuje tudi slika 2. Potencial za premostitev tega prepada bi lahko predstavljal neodvisen standard (na primer OASIS TOSCA), ki bi bil zmožen povezati obe strani in preseči vrzel tehnološke in platformne odvisnosti [24].

Kljub temu bo vedno potrebno upoštevati specifike izbranega ponudnika. Orodja za orkestracijo so sicer zmožna zakriti posebnosti oblaka: med seboj različne oblačne storitve namreč uporabniku lahko predstavijo kot skoraj identične. Še vedno navadno uporabnik doda implementacijo avtomatiziranih nalog, preden se loti orkestracije. Neodvisnost od platforme in ponudnika je ključna in lahko prinese poglavitne prednosti v namestitvi informacijskih rešitev, orkestracija in avtomatizacija

pa lahko razvijalcem dajeta učinkovito podporo pri doseganju le tega.

Naslednji izziv predstavlja odločitev, katero orodje za avtomatizacijo sploh uporabiti. Vsako orodje ima svojo skupnost uporabnikov, ki ga zagovarja, vsako ima svoje prednosti in slabosti, a manjka neodvisna in objektivna primerjava orodij za uporabo v specifičnih okoliščinah in scenarijih. Če izbrano orodje za avtomatizacijo ni optimalno, to sicer zahteva določene prilagoditve pri delu, vendar navadno ne predstavlja usodne napake. Večji vpliv ima izbira orodja za orkestracijo, saj optimalni orkestrator skrije nepotrebne podrobnosti in omogoča fokus zgolj na implementacijo oziroma izbiro avtomatiziranih nalog, nad katerimi se izvaja orkestracija. Razmestitvenih orodij je na pretek, a iz njihovih opisov novinec na tem področju težko razume, kaj sploh je orkestrator in kaj bi moral ta početi. Marsikje je meja med orodji za avtomatizacijo in tistimi za orkestracijo zabrisana in tako je izbira še težja. Pogrešamo objektivne primerjave med različnimi orodji za orkestracijo, pojavlja se tudi dilema, ali dati prednost orodjem, ki delujejo po nekem standardu. Uporaba standarda gotovo prinaša določene prednosti, a lahko pomeni tudi strmo učno krivuljo, saj bo potrebno poleg orodja podrobno razumeti tudi standard za njim.

Po izbiri orodij za avtomatizacijo uporabnika lahko doleti težava, da izbrano orodje ne podpira v celoti naloge, ki si jo je zamislil. Razvijalec tako stoji pred dilemo, ali zamenjati avtomatizacijsko orodje za drugo, manj optimalno, ki podpira manjkajočo funkcionalnost, ali razviti manjkajočo funkcionalnost in jo prispevati v izbrano orodje. Slednja možnost lahko od razvoja modula do potrditve in vključitve v uradno distribucijo traja dlje, kot je sprejemljivo. Odločitev za zamenjavo orodja za avtomatizacijo lahko za sabo potegne tudi nezdružljivost novega orodja z orkestratorjem in posledično tudi menjavo orkestratorja. Orkestratorji so namreč pogosto vezani na določeno orodje za avtomatizacijo. Vse to lahko dodatno otežuje proces razvoja in prinaša zakasnitve v projektu.

7 Zaključek

Področje DevOps želi zmanjšati vrzel med razvojem in namestitvijo ter operativnim delovanjem sodobnih informacijskih rešitev. Prikazali smo značilnosti te prakse in navedli prednosti, pomanjkljivosti in izzive. Definirali in opisali smo procesa avtomatizacije in orkestracije in predstavili njune glavne vidike in vlogo pri razvoju storitev in aplikacij. Pri tem smo nakazali, kaj vse je danes v sodobnem razvoju drugače, kot je bilo pri klasičnem razvoju aplikacij. Avtomatizacija, orkestracija in računalništvo v oblaku tako lahko razumemo kot vznemirljiva področja prihodnosti, ki pa so zelo široka in raznolika, se hitro razvijajo in predstavljajo tudi za izkušene informacijske strokovnjake kopico novih izzivov.

Literatura

[1] C. Ebert, G. Gallardo, J. Hernantes, N. Serrano, *DevOps*, IEEE Software, 33(3), 94-100, 2022.

[2] L. Leite,C. Rocha, F. Kon, D. Milojicic, P. Meirelles (2019). *A survey of DevOps concepts and challenges.*, ACM Computing Surveys (CSUR), 52(6), 1-35.

[3] *The Business Benefits of Automation and Orchestration White Paper*, Cisco 2022, https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/network-services-orchestrator/white-paper-c11-738289.html, dostopano jan. 2022.

[4] R. Kotecha, *9 Cloud Migration Challenges and Solutions For CTOs in 2022*, Simform 2021, https://www.simform.com/blog/cloud-migration-challenges/, dostopano jan. 2022.

[5] *A Business Leader's Guide to DevOps*, 3pillar 2021, https://pages.3pillarglobal.com/a-business-leaders-guide-to-devops.html, dostopano jan. 2022.

[6] M. Gall, F. Pigni, (2021). *Taking DevOps mainstream: a critical review and conceptual framework*, European Journal of Information Systems, 1-20.

[7] T. Delaet, W. Joosen and B. Vanbrabant, *A survey of system configuration tools*,Proc. Large Installations Systems Administration (LISA) conference pp.1-14; Usenix Assoc. 2010.

[8] M. Lenaršič, *Model za ocenjevanje sprejetosti in zrelosti aktivnosti pristopa DevOps*, 2019, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, https://repozitorij.uni-lj.si/Dokument.php?id=123074.

[9] *What is DevSecOps?*, Redhat 2018, https://www.redhat.com/en/topics/devops/what-is-devsecops, dostopano dec. 2021.

[10] R. N. Rajapakse, M. Zahedi, M. A. Babar, H. Shen, H. (2022). *Challenges and solutions when adopting DevSecOps: A systematic review*, Information and Software Technology, 141, 106700.

[11] D. Anjaria, M. Kulkarni (2021). *Effective DevSecOps Implementation: A Systematic Literature Review.*, REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS, 11(4), 4931-4945.

[12] M. P. Groover. *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems.* Wiley 2014.

[13] J. Rifkin, 1995. *The End of Work: The Decline of the Global Labor Force and the Dawn of the Post-Market Era.* Putnam Publishing Group. pp. 66, 75. ISBN 978-0-87477-779-6.

[14] E. Keen, *Automation vs. Orchestration: What's the Difference?*, 2019, https://www.burwood.com/blog-archive/automation-vs-orchestration-whats-the-difference, dostopano dec. 2021.

[15] R. T. Yarlagadda, (2021). *DevOps and Its Practices.* International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882.

[16] I. Salemme, *Pros and Cons of Process Automation*, 2018, https://www.pipefy.com/blog/pros-cons-process-automation/, dostopano dec. 2021.

[17] O. Tomarchio, D. Calcaterra, G. D. Modica, (2020). *PCloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks.* Journal of Cloud Computing, 9(1), 1-24.

[18] J. Gruden, *Avtomatizacija, programabilnost in orkestracija v IT-infrastrukturi*, Revija SRC.SI, junij 2017.

[19] J. Gorišek, *Orkestracija virtualiziranih omrežnih funkcij*, 2016, Fakulteta za elektrotehniko, Univerza v Ljubljani, https://repozitorij.uni-lj.si/Dokument.php?id=88281.

[20] *What Is IT Orchestration (And Why Should You Care)?*, 2017, https://ayehu.com/what-is-it-orchestration-and-why-should-you-care/, dostopano dec. 2021.

[21] M. Caballer, S. Zala, A. Lopez Garcia, G. Molto, P. Orviz Fernandez and M. Velten, *Orchestrating complex application architectures in heterogeneous clouds*, Journal of Grid Computing, Vol. 16, 2017, pp 3-18.

[22] S. Watts, *IT Automation vs Orchestration: What's The Difference*, Business of IT blog, 2020, https://www.bmc.com/blogs/it-orchestration-vs-automation-whats-the-difference/, dostopano: jan. 2022.

[23] C. Stetson, *Principles of Modern Application Development*, 2018, https://www.nginx.com/blog/principles-of-modern-application-development/, dostopano jan. 2022.

[24] J. Wettinger, U. Breitenbücher, F. Leymann. *Standards-based DevOps automation and integration using TOSCA.*, In 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (pp. 59-68). IEEE, 2014.
[25] S. Carey, *Complexity is killing software developers*, Infoworld, 2021, https://www.infoworld.com/article/3639050/complexity-is-killing-software-developers.html.
[26] M. Raza, S. Wickramasinghe, *Automation In DevOps: Why And How To Automate DevOps Practices*, Business of IT blog, 2021, https://www.bmc.com/blogs/automation-in-devops/, dostopano jan. 2022.
[27] B. Lovejoy, *The pros and cons of cloud orchestration*, 2015, https://community.hpe.com/t5/AI-Insights/The-pros-and-cons-of-cloud-orchestration/ba-p/6801095 dostopano dec. 2021
[28] J. Goldberg, *Workflow Orchestration: An Introduction*, 2019, https://www.bmc.com/blogs/workflow-orchestration/, dostopano dec. 2021.
[29] R. Ravulur, *5 ways to modernize your legacy applications*, 2019, https://techbeacon.com/app-dev-testing/5-ways-modernize-your-legacy-applications, dostopano jan. 2022.
[30] C. Stetson, *Principles of Modern Application Development*, 2018, https://www.nginx.com/blog/principles-of-modern-application-development/, dostopano dec. 2021.
[31] T. Abubakr, *Cloud app vs. web app: Understanding the differences*, 2012, https://www.techrepublic.com/blog/the-enterprise-cloud/cloud-app-vs-web-app-understanding-the-differences/, dostopano: dec. 2021.

Anže Luzar je diplomiral na FRI l. 2020 in je trenutno vpisan na magistrski študij.

Mojca Ciglarič je diplomirala (1993) , magistrirala (1996) in doktorirala (2003) s področja računalništva na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Je dekanja FRI in vodja Laboratorija za računalniške komunikacije. Poučuje vsebine s področja računalniških komunikacij in protokolov. Vodila je številne raziskovalne in aplikativne projekte.