Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/20BE47Indjic

TERRAFORM ILI PULUMI - POREĐENJE INFRASTRUKTURNIH ALATA

TERRAFORM OR PULUMI – INFRASTRUCTURE AS CODE TOOLS COMPARISON

Ivan Inđić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Cloud računarstvo postaje sve popularnije iz godine u godinu. Veština pravljenja održive infrastrukture nikada nije bila traženija nego što je sada. Da bi bilo moguće automatizovati i standardizovati proces pravljenja infrastrukture, pojavili su se IaC alati. Infrastruktura kao kod (IaC) je upravljanje i obezbeđivanje infrastrukture putem koda umesto ručnih procesa. IaC alatima se kreiraju konfiguracione datoteke koje sadrže infrastrukturne specifikacije što olakšava uređivanje i distribuciju konfiguracija. Takođe, osigurava da se svaki put kreira identično okruženje i identični resursi. U ovoj oblasti DevOps-a najveći rivali su Terraform i Pulumi.*

Ključne reči: *Infrastruktura, cloud, IaC, Terraform, Pulumi*

Abstract – *Cloud computing is becoming more and more popular every year. The skill of building reusable infrastructure has never been more in demand than it is now. In order to be able to automate and standardize the infrastructure creation process, IaC tools appeared. Infrastructure as Code (IaC) is the management and provisioning of infrastructure through code instead of manual processes. With IaC, configuration files are created that contains infrastructure specifications, making it easy to edit and distribute the configuration. It also ensures that it creates an identical environment and identical resources every time. In this area of DevOps, the biggest rivals are Terraform and Pulumi.*

Keywords: *Infrastructure, Cloud, Iac, Terraform, Pulumi*

1. UVOD

Cloud Computing je model koji omogućava korišćenje resursa koji nisu lokalno prisutni kao što su: serveri, mreže, skladišta podataka, usluge i aplikacije. Pristup ovim resursima je krajnje jednostavan, da bi se koristili potrebna je internet konekcija i bankovna kartica jer se korišćenje resursa plaća.

Postoji tri načina kreiranja infrastrukture. Prvi način je korišćenjem web interfejsa. Drugi način je pomoću CLI komandi. Treći i najpoželjniji način jeste upotrebom IaC alata koji se oslanjaju na API pozive.

Problemi koje IaC alati rešavaju jesu: problemi konzistentnosti infrastrukture, problemi brzine razvoja infrastrukture, problemi kreiranje infrastrukture koja je ponovno iskoristiva i minimizacija ljudskih grešaka.

2. TIPOVI CLOUD RAČUNARSTVA

Javni Cloud Computing (eng. Public Computing) – Sva infrastruktura se nalazi u rukama *cloud* provajdera. Ovo je najbolje rešenje za one koji ne žele da investiraju u IT infrastrukturu. Resursi se dele između svih onih koji koriste resurse. Korisnici plaćaju samo ono što koriste. Činjenica da svi mogu da pristupaju resursima ne znači da je sigurnost narušena u bilo kom smislu. Svi vodeći CSP ulažu velike napore da obezbede visok nivo sigurnosti [1].

Hibridni Cloud Computing (eng. Hybrid Cloud) - Odnosi se na vrstu okruženja koje je sačinjeno od lokalnh resursa, privatnog računarstva u oblaku i javnog računarstva u oblaku. Ovaj oblik računarstva u oblaku omogućava organizaciji da svoje privatne proračune i servise pokreće privatno na svom hardveru, a manje osetljive servise javno. Korišćenje ovakvog okruženja donosi mnoge benefite [2].

Privatni Cloud Computing (eng. Private Cloud) – Obično se nalazi iza zaštitnog zida (eng. firewall) zbog stalnih sigurnosnih pretnji koje vrebaju sa interneta. Uglavnom ga koristi samo jedna organizacija (svi resursi su u vlasništvu jedne organizacije) i može se nalaziti na privatnom ili iznajmljenom hardveru. To znači da organizacija ima sva ovašćenja da menja, konfiguriše i podešava sve što poželi. Resursima ovog tipa računarstva mogu da pristupaju samo one osobe koje su autentifikovane. To pruža veći nivo sigurnosti i veću kontrolu nad infrastrukturom. Ono što se može ispostaviti kao loša strana jesu troškovi održavanja infrastrukture kao i potreba za ekspertima koji će je održavati i obezbediti da ona funkcioniše u skladu sa njenom specifikacijom. Postoji više vrsta ovog tipa *cloud* računarstva [3].

3. SLOJEVI CLOUD RAČUNARSTVA

Infrastruktura kao servis (IaaS) – Predstavlja najniži sloj piramide i služi kao temelj *cloud* računarstva. IaaC je najopsežnija i najfleksibilnija dostupna vrsta usluge. U osnovi, pruža potpuno virtuelizovanu računarsku infrastrukturu kojom se upravlja preko Interneta. IaaC provajder održava fizičku infrastrukturu (serveri i prostor za skladištenje podataka) u data centru, ali omogućava korisnicima da u potpunosti prilagode te virtuelizovane resurse u skladu sa svojim specifičnim potrebama.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila doc. dr Dunja Vrbaški.

Platforma kao servis (PaaS) – Nalazi se na sloju iznad IaaC sloja. Iako IaaC isporučuje celokupnu infrastrukturu i prepušta kupcima da izrade ono što odgovara njihovim potrebama, PaaS je specijalizovaniji. Umesto čiste infrastrukture, PaaS pruža okvir (eng. framework) potreban za izgradnju, testiranje, isporuku, upravljanje i ažuriranje softverskih proizvoda. Koristi istu osnovnu infrastrukturu kao IaaC, ali uključuje i operativne sisteme, razvojne alate i sisteme za upravljanje bazama podataka potrebne za kreiranje softverskih aplikacija.

Softver kao servis (SaaS) - Za većinu ljudi SaaS je najpoznatiji oblik *cloud* računarstva. Smešten na nivou piramide koji je iznad PaaS sloja. SaaS je potpuno razvijeno softversko rešenje spremno za upotrebu putem Interneta na osnovu pretplate. Provajder SaaS-a upravlja: infrastrukturom, operativnim sistemima, posredničkim softverom i podacima neophodnim za isporuku programa, osiguravajući dostupnost softvera kad god i gde god je kupcima potreban. Mnoge SaaS aplikacije rade direktno kroz web pregledače, eliminišući potrebu za preuzimanjima ili instalacijama. Ovo u velikoj meri smanjuje probleme sa upravljanjem softverom za interne IT timove i omogućava kompanijama da pojednostave svoje poslovanje hibridnim i višestrukim oblacima. Neki od primera SaaS-a su: Microsoft Office 365, Salesforce, Google Apps, Trello i slično.

Funkcija kao servis (FaaS) – Ovaj sloj nalazi se u samom vrhu piramide. Ovaj sloj se još naziva i serverles. FaaS deli vrh piramide sa CaaS tehnologijom o kojoj će biti reči kasnije. FaaS omogućava svojim korisnicima da izvršavaju logiku upakovnu u funkcije bez prethodnog obezbeđivanja infrastrukture i servera. *Cloud* provajder obezbeđuje celokupnu infrastrukturu i okruženje koje je potrebno da se funkcija izvrši. Na korisniku je samo da obezbedi izvorni kod funkcije. Funkcije se automatski skaliraju, što ih čini dobro prilagođenim za dinamička opterećenja koja variraju u smislu potrošnje resursa. Primarni nedostatak FaaS-a je vreme izvršavanja. Budući da funkcije treba da obezbede resurse svaki put kada se pokrenu, može doći do primetnih kašnjenja i pada performansi ako aplikacija zahteva puno računarske snage. Primer ovakvih tehnologija su: AWS Lambda, Azure Functions i Google Functions [4].

Kontejner kao servis (CaaS) – Tehnologija koja omogućava kreiranje i upravljanje aplikacijama koristeći apstrakciju zasnovanu na kontejnerima. Radi na drugačijem principu od FaaS-a. Od korisnika se traži da napiše funkciju, zatim ta funkcija biva zapakovana kao kontejner i postavljena na server. Za razliku od FaaS pristupa može se obezbediti stalna aktivnost kontejnera i na taj način se može izbeći "cold start" problem tipičan za FaaS. Korišćenje kontejnera u odnosu na tradicionalan način postavljanja aplikacija ima dosta prednosti [5].

4. BENEFITI KORIŠĆENJA IaC ALATA

Brzina

Prva značajna korist koja se dobija pri korišćenju IaC pristupa je brzina. IaC omogućava brzo postavljanje kompletne infrastrukture pokretanjem skripte. To se učini za svako okruženje, od razvoja do produkcije i za kratko vreme podignuta je identična infrastruktura u svim okruženjima. IaC može učiniti čitav životni ciklus razvoja softvera efikasnijim.

Doslednost

Ručni procesi rezultiraju greškama. Ljudi greše. Komunikacija je teška i generalno smo prilično loši u tome. Ručno upravljanje infrastrukturom rezultiraće određenim odstupanjima, koliko god se trudili. IaC rešava taj problem tako što su same konfiguracione datoteke jedini izvor istine. Na taj način garantujete da će se iste konfiguracije primenjivati iznova, bez razlika.

Odgovornost

Obzirom da je moguće verzionisati sve konfiguracione fajlove, postoji mogućnost praćenja promena koje su se desile. Nije potrebno nagađati ko je šta radio i kada.

Manja cena

Jedna od glavnih prednosti IaC tehnologija je, bez sumnje, smanjenje troškova upravljanja infrastrukturom. Upotrebom *cloud* računarstva uz IaC drastično se smanjuju troškovi. To je zato što nema potrebe za: trošenjem novaca na hardver, trošenjem novca na ljude koji će njime upravljati i trošenjem novca na pravljenje ili iznajmljivanje fizičkog prostora za njegovo skladištenje.

Deklarativnost

Poželjni pristup za IaC jeste upotreba deklarativnih fajlova kada god je to moguće. Deklarativni fajlovi specificiraju šta okruženje zahteva, ali ne nužno i kako. Na primer, moguće je definisati koje komponente su neophodne, ali se eksplicitno ne specificira način i proces instaliranja tih komponenti. Ne postoji standardna sintaksa za pisanje ovakvih fajlova za IaC. Različite platforme podržavaju različite deklarativne formate: JSON, YAML, XML i drugi [6].

5. IZAZOVI U KORIŠĆENJU IaC ALATA

Serverska iskorišćenost (Server Sprawl)

Serverska iskorišćenost se dešava kada data centar ima više servera koji se ne koriste punim kapacitetom. Zbog toga se cena resursa vezanih za memoriju i procesorsku moć ne može opravdati. *Cloud* i virtuelizacija mogu učiniti trivijalnim obezbeđivanje novih servera i resursa. Ovo može dovesti do toga da broj servera raste brže od sposobnosti tima da njima upravlja onako kako bi želeo. Kada se to dogodi, timovi se bore da održe servere ažurnim, potencijalno ostavljajući sisteme ranjivim na poznate eksploatacije. Kada se otkriju problemi, popravke možda neće biti uvedene na sve sisteme na koje one mogu uticati. Razlike u verzijama i konfiguracijama između servera znače da softver i skripte koji rade na nekim mašinama ne rade na drugim.

Izmene u konfiguraciji (Configuration Drift)

Čak i kada se serveri kreiraju na isti način i dosledno konfigurišu, razlike u konfiguraciji se vremenom mogu dogoditi. Na primer, neko ad-hoc konfiguriše jedan od Oracle servera da bi rešio problem koji neko od korisnika ima, i sada se taj server razlikuje od ostalih Oracle servera.

Serveri pahuljice (Snowflake Servers)

Korišćenje IaC tehonologija može pomoći u uklanjanju takozvanih servera pahuljica. Pahuljica server je poseban server na mreži koji se ne može replicirati i ne može se dodirnuti bez problema. To je server koji je izgrađen kao Jenga toranj: tokom dugog perioda detaljno je napravljen da radi na pravi način. Želite da ažurirate na Java 13? Zaboravite na to, ne možete da dodirnete taj server, inače će se cela stvar pokvariti. Previše ovih servera dovodi do krhke infrastrukture. Korišćenje IaC-a znatno olakšava uklanjanje servera i postavljanje novog na njegovo mesto. Korišćenjem IaC-a dobija se definicija koja se automatski primenjuje kada se server obezbedi ili konfiguriše [7].

6. TERRAFORM

Terraform je tehnologija za svakoga ko želi da konfiguriše i upravlja nekom infrastrukturom kao običnim kodom. Infrastruktura se primarno odnosi na *cloud* baziranu infrastrukturu, iako je infrastruktura tehnički sve ono čime može biti upravljno kroz API. Osnovni princip Terraform tehnologije je da omogući pisanje čitljivog konfiguracijskog koda za definisanje IaC-a. Jezik koji koristi Terraform naziva se HashiCorp jezik za konfiguraciju (HCL). Namenjen je uspostavljanju ravnoteže između čitljivog programskog koda, kao i prilagođenosti mašini. Zbog prilagođenosti mašini, Terraform takođe može čitati JSON konfiguracije.

7. PULUMI

Pulumi je moderan IaC alat. Koristi postojeće programske jezike opšte namene kao što su: TypeScript, JavaScript, Python, Go, .Net i Java. Pulumi programi se nalaze u projektu, koji je direktorijum koji sadrži izvorni kod za program i metapodatke o tome kako se program pokreće. Nakon što se napiše program, pokreće se Pulumi CLI komanda *pulumi up* iz direktorijuma projekta. Ova komanda kreira izolovanu i konfigurabilnu instancu programa, poznatu kao stek (eng. stack). Stekovi su slični različitim okruženjima za isporuku koja se koriste prilikom testiranja i postavljanja aplikacija (development, staging, production).

Pulumi koristi model „*desired state“* za upravljanje infrastrukturom. Mehanizam za primenu (eng. deployment engine) upoređuje željeno stanje sa trenutnim stanjem steka i određuje koje resurse treba kreirati, ažurirati ili izbrisati. Mehanizam za primenu infrastrukture koristi skup dobavljača resursa, kao što su: AWS, Azure ili Kubernetes, da bi upravljao pojedinačnim resursima. Dok radi, mehanizam za primenu ažurira stanje infrastrukture informacijama o svim resursima koji su obezbeđeni kao i o svim operacijama na čekanju.

8. POREĐENJE

Nije lako postaviti kriterijume poređenja. Ono što neko vidi kao manu, ne znači nužno da to jeste mana. Poređenje Terraform i Pulumi tehnologija je izvršeno na nekoliko objektivnih kategorija.

1. Lakoća upotrebe
2. Zajednica
3. *Enterprise* ponude za kompanije i timove
4. Mogućnost i lakoća testiranja infrastrukturnog koda

8.1. Lakoća upotrebe

Pretpostavimo da je potrebno da inženjer napiše infrastrukturu za AWS. Takođe, pretpostavimo da do sada nije imao nikakvo iskustvo sa IaC alatima ili nekim drugim jezicima opšte namene (Python, JavaScript, Golang i slično).

Terraform koristi Hashicorp Configuration Language. Ovo je deskriptivni jezik koji ima usku namenu. Ta namena jeste pisanje konfiguracionih fajlova. HCL je zamišljen da bude čitljiv onima koji ga pišu i vrlo je jednostavan. Sastoji od blokova i argumenata. Inženjeru neće biti potrebno puno vremena da se u potpunosti savlada HCL i način na koji se piše modularan i ponovno upotrebljiv Terraform kod.

Kod Pulumi tehnologije je drugačija situacija. On se oslanja na jezike opšte namene. Ovo nudi mnogo veću fleksibilnost i mogućnosti ukoliko je to potrebno. Moguće je modelovati mnogo kompleksnije veze između resursa koji se kreiraju. Inženjer nije obavezan da poznaje programski jezik da bi napisao jednostavnu infrastrukturu. Kada projekat postane kompleksniji, mogućnost dobrog modeliranja veza, odnosa između resursa, kao i mogularizacija resursa postaje ključna. Imajući ovo u vidu, za kvalitetno napisanu kompleksnu infrastrukturu mora se poznavati jezik u kom se ta infrastruktura piše. Ovo povlači veliku količinu utrošenog vremena na učenje principa i načina funkcionisanja nekog jezika opšte namene. Ovaj *overhead* ne postoji u slučaju korišćenja Terraform tehnologije.

8.2 Zajednica

Žašto je zajednica jedan od kriterijuma poređenja? U radu sa IaC alatima, kao i sa svim drugim tehnologijama, neminovne su greške i problemi. Najbolje je, ukoliko je to moguće, učiti iz tuđih grešaka. Ukoliko je zajednica velika, velika je šansa da niste prvi kome se neka greška dogodila i možete iskoristiti poznati način rešavanja te greške.

Stanje zvaničnog Terraform projekta na Github platformi je vrlo dinamično. Trenutno postoji 1654 ljudi koji svojim radom doprinose razvoju. U vreme pisanja ovog rada, postoji ~34.000 zvezdica i ~ 8.000 *fork*-ova. Zvezdice imaju dvostruku namenu. Neki ljudi koriste zvezdice da naznače da im se projekat sviđa, drugi ih koriste kao obeleživače kako bi kasnije mogli da prate šta se dešava na repozitorijumu. Trenutno postoji 141 *pull request*-ova i 1.500 *issue*-a.

Pulumi poslednjih godina značajno napreduje u pogledu broja podržanih provajdera i ljudi koji doprinose tome. U vreme pisanja ovog rada, postoji 178 ljudi koji doprinose razvoju Pulumi-ja. Aktuelna verzija je 3.38.0, a ukupno postoji 122 *release*. Vreme izbacivanja novog *release*-a je slično kao i u Terraformu-u, 2 do 3 nedelje. Sudeći po Github projektima, Pulumi ne uživa toliku popularnost kao njegov suparnik. Trenutno ima: ~15000 zvezdica, ~750 *fork*-ova, 90 *pull request*-ova i 1300 *issue*-a. Ono gde Pulumi blista je javni Slack kanal gde je moguće pitati ukoliko postoji neki problem.

Oba alata imaju *enterprise* verziju koja nije besplatna i omogućava korišćenje cloud upravljanje verzijama ovih alata. Počnimo od Terraform-a.

Zašto bi neko koristio Terraform enterprise verziju?

1. Ušteda vremena na standardizaciji *deployment*-a. Moguće je na lak način kreirati i distribuirati module i provajdere unutar organizacije koristeći privatne registre.

2. Jednostavna CICD integracija. Jednostavno je povezati VCS sa Terraform cloud-om i dodati korak gde je potrebno ručno potvrditi deployment infrastrukture.

3. Evaluator troškova. Za svaku promenu u infrastrukturi moguće je dobiti tačnu informaciju o promeni troškova koja će se desiti ukoliko se infrastruktura promeni.

4. Podrška koju Terraform omogućava svojim klijentima je moguća putem web portala ili elektronske pošte. Za većinu tehničkih problema, HashiCorp podrška će tražiti dijagnostičke informacije uključene u zahtev za podršku. Da bi obezbedio da su potrebne informacije uključene, Terraform Cloud može automatski da generiše paket podrške (eng. support bundle) uključujući logove i detalje o konfiguraciji.

Pulumi takođe ima *enterprise* verziju. Neke od benefita koje *enterprise* omogućava su:

1. Zaštita polisama. Sprovođenje polisa širom organizacije na strani servera, uključujući: najbolje prakse za usklađenost i bezbednost, ograničenja pristupa mreži i kontrolu troškova.

2. Upravljanje federalnim identitetom i grupama. Organizacije mogu da donesu sopstveni SAML 2.0 i jedinstvenu prijavu (SSO) za upravljanje identitetom u celom preduzeću i kontrolama pristupa zasnovanim na ulogama.

3. Nadzor. Pulumi će evidentirati sve aktivnosti koje su korisnici uradili, uz mogućnost eksportovanja evidencije i integracije sa drugim alatima za bezbednost.

4. Hostovanje kod klijenta. Preduzeća koja zahtevaju specifične kontrole podataka mogu samostalno da hostuju Pulumi u svom privatnom *cloud*-u ili data centru, zajedno sa: izolacijom mreže, identitetom i kontrolama vlasništva podataka.

8.4 Mogućnost i lakoća testiranja infrastrukturnog koda

Oba alata podrzavaju testiranje infrastrukture. Terraform podržava: jedinične, ugovorne, integracione i end-to-end testove. Pulumi podržava: jedinično, integraciono i testiranje svojstava. S obzirom da Pulumi koristi jezike opšte namene, testiranje se oslanja na okvire za testiranje tih jezika. Ovo znači da testiranje deluje vrlo prirodno dok u slučaju Terraform-a to nije slučaj.

9. ZAKLJUČAK

Na kraju krajeva, odabir najboljeg alata zavisi od potreba. Iz ličnog iskustva mogu reći da je Terraform mnogo više korišćen na komercijalnim, klijentskim projektima. Terraform je više prihvaćen i od strane *DevOps*-a a čak i od strane programera koji već poznaju Python ili neki drugi jezik opšte namene. Ima više razloga koji dovode do ovoga. Kada pričamo od održavanju infrastrukture, Terraform je čist pobednik. Ne postoji uniforman način niti standard u pisanju infrastrukture koristeći Pulumi i jezik opšte namene. Ovo je posledica velike fleksibilnosti koje nude jezici opšte namene.

Kada imamo 10 inženjera koji pišu istu infrastrukturu u jeziku Python, oni to mogu uraditi na 10 različitih načina. Sve ovo dovodi do problema u održavanju infrastrukturnog koda. Kada na projekat pisan u Pulumi-ju dođe inženjer koji ne poznaje taj jezik ili stil i način pisanja, njegov *onbarding* proces će trajati jako dugo. Trajanje ovog procesa proporcijalno raste sa kompleksnošću projekta.

Sa druge strane, kada pričamo o Terraform-u, postoji jedan i samo jedan ispravan način da se napiše kvalitetna infrastruktura.

10. LITERATURA

[1] https://munzandmore.com/2015/ora/12-public-cloud-benefits-and-features-you-should-know (pristupljeno u septembru 2022.)

[2] https://searchcloudcomputing.techtarget.com/definition/hybrid-cloud (pristupljeno u septembru 2022.)

[3] https://www.sciencedirect.com/topics/computer-science/private-cloud (pristupljeno u septembru 2022.)

[4] https://www.vxchnge.com/blog/different-types-of-cloud-computing (pristupljeno u septembru 2022.)

[5] https://www.redhat.com/en/topics/cloud-computing/what-is-caas (pristupljeno u septembru 2022.)

[6] https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/ (pristupljeno u septembru 2022.)

[7] https://www.oreilly.com/library/view/infrastructure-as-code/9781491924334/ch01.html (pristupljeno u septembru 2022.)

Kratka biografija:

Ivan Indić rođen je u Beogradu 16.03.1996 godine. Upisao Fakultet tehničkih nauka u Novom Sadu 2015. godine, smer Elektrotehnika i računarstvo. Diplomirao 2020. godine sa temom „Case-basedreasoning“. Nakon osnovnih studija upisuje master studije na istom fakultetu. U slobodno vreme voli da igra šah i putuje.