
Računalstvo u oblaku II

Orkestracija i DevOps

Poglavlje 12 — Skripta kolegija *Računalne mreže*

doc. dr. sc. Sandi Baressi Šegota

Fakultet informatike Sveučilišta Jurja Dobrile u Puli

Poglavlje 12

Računalstvo u oblaku II: orkestracija i DevOps

Kad broj kontejnera naraste sa nekolicine na stotine, njima više nije moguće upravljati ručno. Ovo poglavlje bavi se *orkestracijom* kontejnera (Kubernetes), *infrastrukturom kao kodom* (IaC), praksama CI/CD i DevOps kulture te osnovama mreža u oblaku (VPC, podmreže, balanseri). Te tehnologije zajedno omogućuju pouzdano, ponovljivo i automatizirano upravljanje modernim aplikacijama.

Ishodi učenja

Po završetku poglavlja student će moći: objasniti potrebu za orkestracijom i opisati arhitekturu i objekte Kubernetesa; objasniti infrastrukturu kao kod; opisati CI/CD cjevovod i DevOps kulturu; te objasniti VPC i mrežne usluge oblaka.

12.1 Orkestracija i Kubernetes

Pokretanje jednog kontejnera je jednostavno, no u produkciji ih ima mnogo, treba ih raspoređivati na poslužitelje, skalirati prema opterećenju te automatski oporavljati nakon ispada. Taj posao obavlja **orkestrator**, a de facto standard je **Kubernetes**: sustav koji automatizira postavljanje, skaliranje i upravljanje kontejneriziranim aplikacijama. Korisnik opisuje *željeno stanje*, a Kubernetes ga neprestano održava.



Figure 12.1: Orkestrator koordinira pokretanje, skaliranje i oporavak mnoštva kontejnera.

Kubernetes klaster dijeli se na **upravljačku ravninu** — s *API poslužiteljem* (ulazna točka), *raspoređivačem* (dodjeljuje podove čvorovima), pohranom stanja *etcd* i kontrolerima — te **radne čvorove** na kojima *kubelet* pokreće podove.

Temeljni objekti su: **Pod** (najmanja jedinica, jedan ili više kontejnera), **Deployment** (održava željeni broj replika podova i vodi ažuriranja) te **Service** (daje stabilnu pristupnu točku i raspoređuje promet na podove). Skaliranje se svodi na promjenu broja replika, a Kubernetes ga može i *automatizirati* prema metrikama. Zahvaljujući stalnom težnji željenom stanju, sustav se *samoizlječuje*: ako pod padne, kontroler automatski stvara novi.

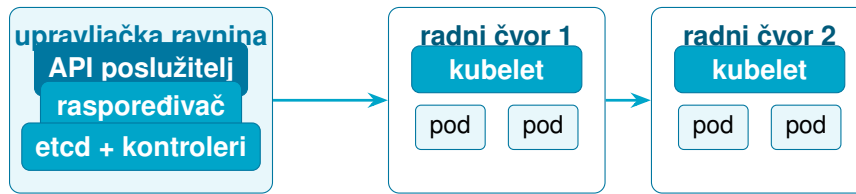


Figure 12.2: Arhitektura Kubernetesa: upravljačka ravnina i radni čvorovi.

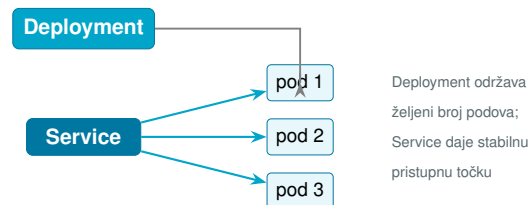


Figure 12.3: Deployment održava podove, a Service im daje stabilnu pristupnu točku.

Konfiguracija se odvaja od koda kroz *ConfigMap* (postavke) i *Secret* (osjetljivi podaci), a klaster se logički dijeli *namespace*-ovima (npr. po timovima ili okruženjima).

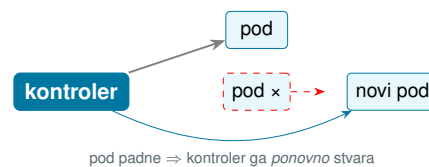


Figure 12.4: Samoizlječenje: kad pod padne, kontroler automatski stvara novi.

12.2 Infrastruktura kao kod

Ručno postavljanje resursa (klikanjem kroz sučelje) sklono je pogreškama, teško se ponavlja i ne ostavlja evidenciju. **Infrastruktura kao kod** (IaC) rješava to opisivanjem infrastrukture *kodom* koji se verzionira i automatski primjenjuje. Pristup je uglavnom *deklarativan*: opisuje se *željeno stanje* (što želimo), a alat sam izračuna korake do njega — za razliku od *imperativnog* pristupa gdje se navode sami koraci.

Popularan alat je **Terraform**: najprije prikazuje *plan* promjena, a nakon potvrde ih *primjenjuje* (apply), pritom vodeći *stanje* infrastrukture. Prednosti IaC-a su ponovljiva i dosljedna okruženja, pregledne promjene (code review i povijest) te brz oporavak i repliciranje. Uz to se često primjenjuje *GitOps*, gdje je Git izvor istine, a promjene prolaze kroz pull request.

12.3 CI/CD i DevOps

DevOps je kultura koja spaja razvoj (*Dev*) i operacije (*Ops*), naglašavajući automatizaciju, suradnju i brzu povratnu spregu radi čestih, malih i pouzdanih isporuka. Tehnički temelj su **CI/CD** cjevovodi. *Kontinuirana integracija* (CI) znači da developeri često spajaju kod, uz automatski build i testove koji rano otkrivaju pogreške. *Kontinuirana isporuka/postavljanje* (CD) nakon uspješnih testova automatski priprema (ili izravno postavlja) aplikaciju u produkciju.

Za sigurne isporuke koriste se strategije poput *plavo-zeleno* (dvije okoline uz brzo prebacivanje) i *kanarinca* (nova verzija najprije za dio korisnika), uz mogućnost brzog *povrata*

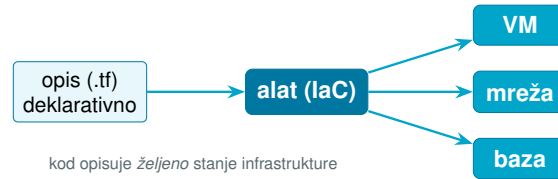


Figure 12.5: IaC: deklarativni opis pretvara se u stvarne resurse u oblaku.

	deklarativno	imperativno
opisuje	željeno stanje	korake
alat	računa razliku	izvršava redom
ponovljivost	visoka	niža
primjer	Terraform	skripte

Table 12.1: Deklarativni i imperativni pristup infrastrukturi.

(rollback) ako se pojavi problem.

12.4 Mreže u oblaku

Resursi u oblaku smještaju se u **VPC** (virtualnu privatnu mrežu) — izoliranu mrežu s vlastitim adresnim prostorom, podijeljenu na *podmreže*. *Javne* podmreže dostupne su s Interneta (npr. za balanser), dok su *privatne* interne (baze, aplikacije); vezu s Internetom omogućuje *gateway*. Pristup se dodatno kontrolira *sigurnosnim grupama* (vatrozidom na razini instance) uz načelo najmanje privilegije.

Balanser opterećenja raspoređuje promet na više instanci uz provjere zdravlja, čime raste dostupnost i propusnost. Servisi se međusobno pronalaze po *imenu* (interni DNS), a sadržaj se može približiti korisniku putem *CDN*-a. Sva ta načela — kontejneri, mreže, balansiranje i reverse proxy — primjenjiva su i na malom, vlastitom poslužitelju, što je most prema sljedećem predavanju o HomeLabu.

Sažetak poglavlja

Orkestracija (Kubernetes) automatizira rad s mnoštvom kontejnera preko objekata *Pod*, *Deployment* i *Service*, uz skaliranje i samoizlječenje. **IaC** opisuje infrastrukturu kodom (deklarativno, ponovljivo; npr. Terraform). **CI/CD** automatizira put od koda do produkcije unutar *DevOps* kulture. Mreže u oblaku organiziraju se kroz **VPC**, podmreže, sigurnosne grupe i balansere.

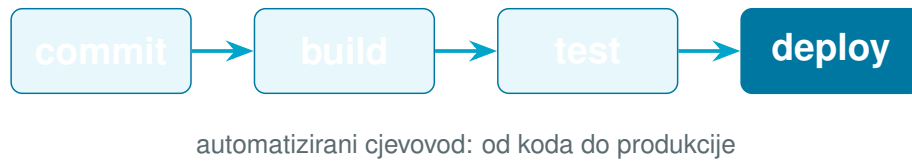


Figure 12.6: CI/CD cjevovod: svaka promjena prolazi isti put od commita do produkcije.



Figure 12.7: VPC s javnom i privatnom pod mrežom te izlazom na Internet.

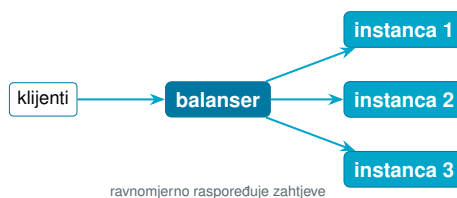


Figure 12.8: Balanser opterećenja ravnomjerno raspoređuje zahtjeve na više instanci.