Razvoj infrastrukture za učenje DevOps razvojnih principa

Marko Mudrinić

***Sadržaj* — Rad prikazuje proces izgradnje infrastrukture za učenje DevOps razvojnih principa na primeru implementacije privatnog oblaka na Računarskom fakultetu. Opisani su korišćeni alati i tehnologije, arhitektura sistema, način upotrebe u nastavi, prednosti i mane rešenja, kao i planirana alternativna rešenja.**

***Ključne reči* — *DevOps*, edukacija iz oblasti softverskog inženjerstva (eng. *software engineering education*), *Kubernetes*, računarstvo u oblaku (eng. *cloud computing*).**

I. Uvod

Uspešno usvajanje *DevOps* razvojnih principa podrazumeva rad sa sistemima za kontrolu verzija, rad sa platformama za kontinualnu integraciju i isporuku, implementaciju kontinualnog praćenja, kao i postavljanje razvijenog projekta u produkciono okruženje.

Svi navedeni zadaci se realizuju korišćenjem odgovarajućih tehnologija, platformi i alata koje je potrebno pokrenuti na odgovarajućim serverima, odnosno infrastrukturi. *DevOps* razvojni principi se primenjuju na projektima na kojima radi veliki broj programera, pa lokalno pokretanje (na računaru programera) potrebnih alata jednostavno nije praktično, niti omogućava da se iskoriste prednosti koje pruža *DevOps* [1].

Kako se *DevOps* razvojni principi sve više koriste u industriji, raste i potreba za obrazovanjem studenata u oblasti *DevOps*-a. Infrastruktura je najčešća prepreka za integraciju *DevOps*-a u nastavni plan. Veliki broj platformi i alata korišćenih u implementaciji *DevOps*-a su prilagođeni okruženjima u oblaku. Fakultetima je pak iznajmljivanje infrastrukture u oblaku veliki trošak koji često ne mogu da priušte.

Marko Mudrinić, Autor, Računarski fakultet, Srbija (telefon: 381-69-1150511; email: mmudrinic @raf.rs)

Ovaj rad pokazuje kako je moguće rešiti ovu prepreku implementacijom privatnog oblaka na sopstvenim serverima fakulteta koristeći tehnologije i platforme koje su besplatne i/ili otvorenog koda. Ovakav privatan oblak se koristi u praksi za učenje *DevOps* razvojnih principa na Računarskom fakultetu Univerziteta Union u Beogradu.

II. KUBERNETES KAO CENTRALNA PLATFORMA

Kubernetes (skraćeno *K8s*) je najpopularnija platforma za orkestraciju (aplikativnih) kontejnera, što uključuje njihovo pokretanje, upravljanje i skaliranje. *Kubernetes* organizuje kontejnere koji pripadaju jednoj aplikaciji u logičke celine, što olakšava upravljanje njima i otkrivanje servisa (eng. *service discovery*).

Postoji mnogo razloga zašto je *Kubernetes* postao toliko popularan i zašto je *de-facto* standard za pokretanje aplikacija u timovima koje koriste *DevOps*. Neki od razloga uključuju to da iza *Kubernetes* projekta stoje najveće svetske kompanije poput Gugla, *Red Hat*-a, Amazona i drugih, ali i to da je *Kubernetes* izuzetno prilagodljiva platforma, te može da se prilagodi raznim okruženjima i potrebama [2].

Ideja je da se *Kubernetes* instalira na serverima fakulteta i koristi kao centralna platforma za pokretanje svih potrebnih servisa, komponenti i studentskih projekata. Razlozi za to su što veliki broj savremenih platformi nativno podržava *Kubernetes* i što on omogućava raspoređivanje kontejnerskih aplikacija tako da se dostupni resursi maksimalno iskoriste uz minimalno rasipanje (tzv. „*bin packing*“).

Prvi i glavni cilj je pružiti studentima *Kubernetes* klastere uslužno kao što to rade dobavljači infrastrukture u oblaku. Kako su fakultetima resursi ograničeni, nije moguće pružiti svakom studentu klaster, ali je moguće pružiti nekoliko klastera po predmetu na kome za to ima potrebe, a studenti te klastere mogu da „dele“ po potrebi.

Drugi cilj je omogućiti studentima da sami upravljaju dodeljenim klasterima, odnosno resursima, kako bi stekli iskustvo u radom sa infrastrukturom sličnoj infrastrukturi u oblaku. To uključuje i inicijalno pravljenje klastera, ali i naknadno održavanje klastera, dodavanje ili smanjenje resursa i broja čvorova u klasteru. Način na koji ovo može da funkcioniše je taj što bi fakultet obezbedio platformu koja bi to omogućila i dovoljnu količinu resursa, a zatim svakom predmetu dodelio određenu količinu resursa na osnovu potreba. Nastavno osoblje i studenti bi zatim samostalno koristili te resurse za pravljenje određenog broja klastera.

Pored studentskih projekata, *Kubernetes* može da se koristi i za pokretanje

svih ostalih platformi i alata potrebnih za učenje *DevOps* razvojnih principa. To uključuje platformu za kontinualnu integraciju i isporuku, platformu za kontinualno praćenje, komponente poput baza podataka, skladišta podataka i slično. Predmeti na koji se obrađuje tema *DevOps*-a mogu da pokrenu sve potrebne komponente tako što će napraviti jedan klaster posvećen tome i podeliti ga sa svim studentima na predmetu. Dodatni klasteri za pokretanje studentskih projekata mogu da se prave za manje grupe studenata u zavisnosti od količine dostupnih, odnosno dodeljenih resursa.

III. ODABIR PLATFORMI ZA REALIZACIJU PRIVATNOG OBLAKA

Postoje razna rešenja za implementaciju privatnog oblaka. Najpopularnija tradicionalna rešenja uključuju *OpenStack*, *Proxmox VE*, *VMware vCloud Director* (odnosno *vSphere* i *vCenter*) i *OpenNebula*. Problem tradicionalnih rešenja, naročito onih otvorenog koda, jeste što njihovo postavljanje i održavanje može biti vrlo komplikovano i vremenski zahtevno.

Važno je napomenuti da fakulteti imaju vrlo ograničene ljudske resurse za upravljanje infrastrukturom. Obično su to nastavno osoblje (profesori i asistenti) koji tu infrastrukturu i koriste. Nastavno osoblje ima i druge obaveze, poput pripreme materijala, organizacije predmeta, rada sa studentima i naučnog rada. Iz tog razloga, izuzetno je bitno naći rešenje koje će omogućiti da se infrastruktura postavi, skalira i održava na vremenski efikasan način, uz minimalan operativni teret i jasne procedure, kako bi glavni fokus ostao na nastavi i podršci studentima. Kako se *Kubernetes* već javlja kao platforma za pokretanje raznih drugih platformi i komponenti, pitanje je kako *Kubernetes* može da se iskoristi za realizaciju privatnog oblaka.

Rešenje koje se nametnulo kao idealno je kombinacija *Kubermatic Kubernetes* platforme sa *KubeVirt* platformom za virtuelizaciju. Obe platforme su besplatne ili nude besplatno izdanje i imaju i više nego dovoljno funkcionalnosti za realizaciju zadatih ciljeva. Druge analizirane platforme nisu imale sve potrebne funkcionalnosti ili su bile komercijalne sa cenama licenci u iznosu od više desetina hiljada dolara godišnje.

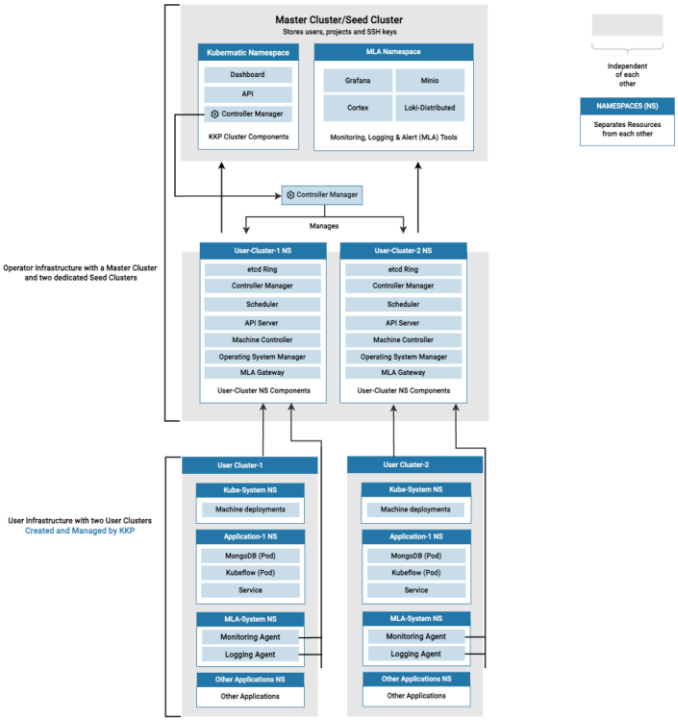
Kubermatic Kubernetes platforma (skraćeno *KKP*) je platforma otvorenog koda za centralizovano upravljanje *Kubernetes* klasterima u okruženjima u oblaku, kao i u *on-premise* i *bare-metal* okruženjima. *KKP* obezbeđuje potpunu automatizaciju životnog ciklusa klastera, što uključuje pravljenje, nadogradnju i održavanje (tzv. "*Day-2 operations*"). Platforma dolazi u dva izdanja: besplatno (tzv. *Community*) i komercijalno (tzv. *Enterprise*). Razlika u ovim izdanjima je to da komercijalno izdanje pruža dodatne

funkcionalnosti potrebne velikim kompanijama koje imaju više stotina klastera.

Kubernetes klasteri napravljeni putem *KKP*-a imaju malo drugačiju arhitekturu i organizaciju u odnosu na tradicionalne *Kubernetes* klastere. *KKP* koristi sličnu arhitekturu kao i dobavljači infrastrukture u oblaku. Umesto da se *control plane* komponente klastera (komponente koje pružaju funkcionalnosti *Kubernetes*-a) napravljenih preko *KKP*-a pokreću na serverima odvojenim za to, *KKP* pokreće ove komponente kao *Pod*-ove (kontejnere) u klasteru namenjenom za to (tzv. "*seed*" klaster). U okviru besplatnog izdanja, moguće je imati jedan *seed* klaster, dok komercijalno izdanje ima podršku za više *seed* klastera. Ovo pruža izuzetno veliku moć skalabilnosti; umesto da se jedan server koristi samo za jedan klaster, sada se jedan server koristi za više klastera, dok *Kubernetes* i *KKP* obezbeđuju potrebnu izolaciju i bezbednost.

Radni čvorovi (eng. *worker nodes*) klastera mogu da budu u nekom od podržanih dobavljača infrastrukture u oblaku, poput *AWS*-a, *Microsoft Azure*-a i *Google Cloud*-a, ali i u nekom od privatnih okruženja zasnovanih na *VMware vSphere*-u, *OpenStack*-u, *KubeVirt*-u ili *Tinkerbell*-u. Ovakva arhitektura omogućava da se *control plane* pokreće na jednom mestu, a da se aplikacije (eng. *workload*) pokreću na više različitih dobavljača i/ili okruženja, što doprinosi visokoj dostupnosti aplikacija.

Na slici 1 prikazana je bazična arhitektura *KKP*-a [3].



Sl. 1. Bazična arhitektura besplatnog izdanja *Kubermatic Kubernetes* platforme [3].

Postavlja se pitanje šta je sa radnim čvorovima. Tu nastupa ranije pomenuta *KubeVirt* platforma, koja je jedno od podržanih okruženja u kome *KKP* može da pravi radne čvorove.

KubeVirt je platforma otvorenog koda koja omogućava upravljanje virtuelnim mašinama u okviru *Kubernetes*-a. Virtuelne mašine napravljene preko *KubeVirt*-a izvršavaju se na radnim čvorovima *Kubernetes* klastera. *KubeVirt* virtuelnim mašinama upravlja koristeći dva najpopularnija alata za to: *KVM* i *QEMU*. Način na koji ovo funkcioniše je taj da *KubeVirt* platforma pruža odgovarajuće *Kubernetes* resurse i kontrolere za upravljanje virtuelnim mašinama. Pravljenjem *Kubernetes* resursa *VirtualMachineInstance*, korisnik dobija jednu virtuelnu mašinu na jednom od radnih čvorova tog klastera [4].

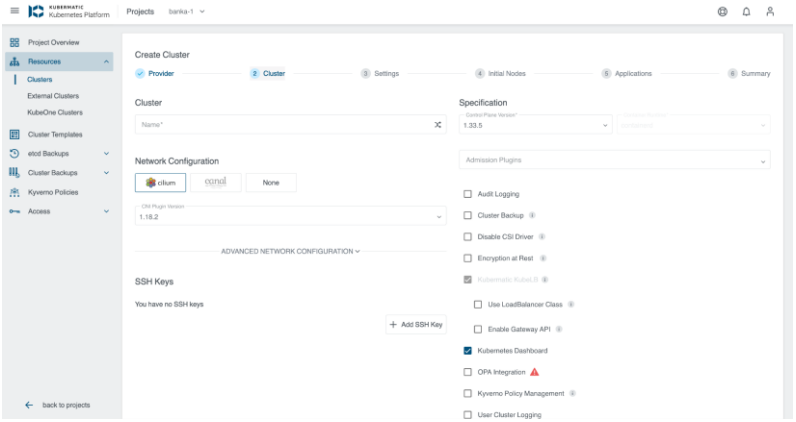
KKP koristi *KubeVirt* na sledeći način: korisnik (u ovom slučaju student) prilikom pravljenja klastera bira *KubeVirt provajdera* i željeni broj radnih čvorova. Nakon toga odgovarajuće komponente *KKP*-a (*machine-controller* i *operating-system-manager*) prave *VirtualMachineInstance* objekte sa potrebnom konfiguracijom. Ta konfiguracija sadrži instrukcije koje

omogućavaju virtuelnim mašinama da se automatski priključe odgovarajućem klasteru kao radni čvor. *KKP* takođe sadrži integraciju sa *cluster-autoscaler* komponentom *Kubernetes*-a, pa virtuelne mašine mogu dinamički da se dodaju i brišu u zavisnosti od opterećenja pojedinačnih klastera. Studenti takođe mogu u bilo kom trenutku samostalno da promene broj virtuelnih mašina, odnosno broj radnih čvorova u njihovim *Kubernetes* klasterima.

Upotreba virtuelnih mašina za radne čvorove omogućava da se resursi bolje raspodele. Na primer, ukoliko za jedan klaster treba više resursa nego za drugi, taj klaster će imati više ili veće virtuelne mašine. Osim toga, virtuelne mašine predstavljaju način da se ograniči koliko maksimalno resursa svaki klaster može da iskoristi, tako da ne dođe do situacije da jedan klaster iskoristi više resursa nego planirano i na taj način ugrozi rad ostalih klastera.

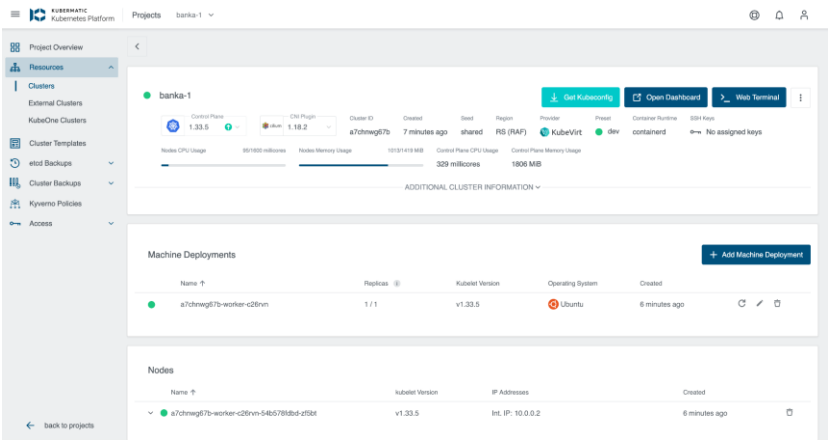
KKP podražava dva načina upravljanja Kubernetes klasterima: korišćenjem grafičkog korisničkog interfejsa i korišćenjem *API*-ja. Kako proces pravljenja klastera sadrži ogroman broj promenljivih, studentima je preporučeno da koriste grafički interfejs [3]. Na slici 2 prikazan je pregled svih klastera u okviru jednog *KKP* projekta.

Na slici 2 prikazana je jedna od formi u procesu pravljenja novog klastera.



Sl. 2. Jedna od formi u procesu pravljenja novog klastera u okviru KKP-a (izvor: snimak ekrana)

Na slici 3 prikazan je pregled pojedinačnog klastera, koji pokazuje status klastera, količinu dostupnih i iskorišćenih resursa, kao i status čvorova (odnosno virtuelnih mašina). Takođe, na ovoj stranici studenti mogu da preuzmu odgovarajući *kubeconfig* fajl koji se koristi za pristup klasteru.



Sl. 3. Pregled jednog klastera napravljenog preko *KKP*-a (izvor: snimak ekrana)

IV. IMPLEMENTACIJA ARHITEKTURE NA RAČUNARSKOM FAKULTETU

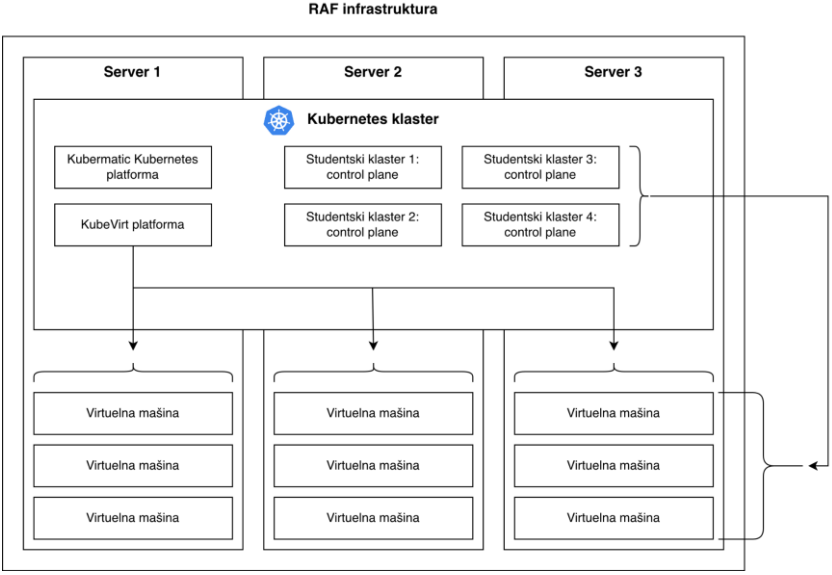
Računarski fakultet je izdvojio za implementaciju ovog rešenja tri servera sa ukupno 96 virtuelnih procesora, 756 gigabajta radne memorije (*RAM*) i 600 gigabajta *SSD* skladišnog prostora. U trenutku pisanja ovog rada, ovo rešenje se koristi za izvođenje nastave na predmetu „Softversko inženjerstvo“ koje za cilj ima da nauči studente da rade u okruženjima koje koriste *DevOps* razvojne principe i *cloud native* tehnologije.

Na dodeljenim serverima je instalirana *Kubernetes* platforma koristeći *Kubermatic KubeOne* alat koji podržava tzv. *bare-metal* servere, prati sve preporuke za klastere u produkcionim okruženjima date od strane *Kubernetes* projekta i omogućava deklarativno definisanje klastera. Jednostavnosti radi, ovaj klaster ćemo zvati „glavni klaster“. Kako bi se što više resursa oslobodilo za studentske klastere, sva tri čvora se koriste za pokretanje aplikacija i virtuelnih mašina, tj. nijedan čvor/server nije odvojen isključivo za *control plane* komponente. Na tom klasteru su instalirane i konfigurisane *Kubermatic Kubernetes* platforma i *KubeVirt* platforma.

U okviru *KKP* platforme se pravi po jedan klaster za svaki razvojni tim na predmetu „Softversko inženjerstvo“, kojih ima četiri. Na glavnom klasteru se pokreću *control plane* komponente za sva četiri studentska klastera, kao i virtuelne mašine koje služe kao radni čvorovi u studentskim klasterima. Svaki studentski klaster ima tri virtuelne mašina sa po četrnaest virtuelnih procesora (eng. *vCPU*) i 96 gigabajta radne memorije.

Na studentskim klasterima se dalje pokreću sve potrebne komponente za primenu *DevOps*-a i studentski projekti.

Slika 4 prikazuje opisanu implementaciju u formi dijagrama.



Sl. 4. Pregled implementacije privatnog oblaka na RAF infrastrukturi

V. MANE ARHITEKTURE I ALTERNATIVNI PRISTUPI

Ono što predstavlja najveću manu u trenutnoj arhitekturi je dodatno opterećenje (eng. *overhead*) koji proizilazi iz upotrebe virtuelnih mašina. Kako virtuelne mašine imaju svoj operativni sistem i pokreću sve propratne servise tog operativnog sistema, potrebno je izdvojiti znatnu količinu resursa samo za potrebe pokretanja virtuelnih mašina.

Druga velika mana je vreme potrebno da se virtuelna mašina napravi, inicijalizuje i pridruži klasteru. Iskustvo je pokazalo da je za to potrebno između 30 minuta i sat vremena, što u nekim slučajevima nije prihvatljivo, npr. kada se pravi nova virtuelna mašina zbog nedostatka resursa za pokretanje neke komponente.

Plan je da se u narednim školskim godinama izvrši evaluacija alternativnih rešenja. Jedno od mogućih alternativnih rešenja su virtuelni *Kubernetes* klasteri. Oni funkcionišu tako što se jedan fizički *Kubernetes* klaster deli na više logičkih klastera. Kada korisnik napravi logički klaster, on može da vidi samo objekte koji pripadaju tom logičkom klasteru. Svi objekti u logičkom klasteru zapravo postoje i u fizičkom klasteru; logički klasteri u suštini ograničavaju šta ko može da vidi. Iz ovoga proizilazi znatno jednostavnija arhitektura. Umesto više instanci *control plane*-a, postoji samo jedan *control*

plane, odnosno *control plane* fizičkog klastera. Takođe, više ne postoji potreba za virtuelnim mašinama, već se direktno koriste čvorovi fizičkog klastera [5].

Postoji više rešenja koja omogućavaju ovaj pristup, a trenutno su najpopularnija *vCluster* i *kcp*. Razlika između ova dva rešenja je u opsegu. *vCluster* se fokusira na pokretanje aplikacija, dok se *kcp* fokusira na deljenje *API*-ja između provajdera tih *API*-ja i korisnika. Drugim rečima, *kcp* nema ugrađenu podršku za pokretanje *Pod*-ova kao što je to slučaj sa *vCluster*-om, već administrator sistema mora sam da napravi komponente koje će da vrše replikaciju *Pod*-ova (i ostalih relevantnih resursa) iz logičkih klastera u fizički klaster [6]. Mana *vCluster*-a je u tome što nije u potpunosti besplatno rešenje. Iako postoji besplatno izdanje, to izdanje ima dosta ograničenja u vidu dostupnih funkcionalnosti [5].

Ovaj pristup bi mogao značajno da smanji potrebnu količinu resursa, dok bi se princip privatnog oblaka i dalje zadržao, samo što bi studenti sada radili sa virtuelnim klasterima, a ne sa virtuelnim mašinama. Ovo je istovremeno mana i problem, jer rad sa virtuelnim klasterima neće pružiti istu mogućnost za edukaciju studenata kao kada studenti rade sa virtuelnim mašinama koje su nalik pravim serverima. Dodatno, virtuelni klasteri ne pružaju dovoljno dobre funkcionalnosti za ograničavanje iskorišćenja resursa u okviru virtuelnih klastera. Iz tog razloga trenutno se razmatra kombinacija ova dva rešenja za virtuelne klastere, zajedno sa rešenjima poput *KKP*-a i *KubeVirt*-a.

VI. Zaključak

Ova arhitektura se u praksi pokazala kao veoma stabilna i efikasna, kako sa tehničkog tako i sa organizacionog aspekta. Sa stanovišta održavanja, rešenje je izuzetno pouzdano i ne zahteva stalnu pažnju ili kompleksne intervencije. Nakon početnog postavljanja sistema i podešavanja osnovnih komponenti, rad platforme je u najvećoj meri autonoman. Osim redovnih ažuriranja, gotovo da nije bilo potrebe za dodatnim tehničkim zahvatima ili manuelnim intervencijama.

Čak i u situacijama kada bi došlo do potpunog prekida rada, na primer zbog nestanka električne energije ili privremenog pada mrežne infrastrukture, sistem bi se samostalno oporavio nakon ponovnog pokretanja servera. Kubernetes automatski detektuje stanje komponenti i obezbeđuje da se svi neophodni servisi ponovo pokrenu bez potrebe za ručnom intervencijom. Na taj način proces oporavka je u potpunosti automatizovan, što značajno olakšava održavanje.

Ovakva implementacija potvrđuje da se kombinacijom različitih rešenja može izgraditi održiva i efikasna infrastruktura pogodna za edukaciju studenata. Arhitektura omogućava studentima da rade u okruženju koje

funkcionalno i organizaciono oponaša realne sisteme u oblaku, čime se ostvaruje značajan doprinos praktičnom učenju *DevOps* principa.

LITERATURA

[1] N. Forsgren, J. Humble and G. Kim, Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations, IT Revolution, 2018.

[2] The Kubernetes Authors, "Kubernetes Overview," [elektronski izvor]. Available: https://kubernetes.io/docs/concepts/overview/. [Poslednji pristup 09 Novembar 2025].

[3] Kubermatic GmbH, "Kubermatic Architecture," [elektronski izvor]. Available: https://docs.kubermatic.com/kubermatic/v2.29/architecture/. [Poslednji pristup 09 Novembar 2025].

[4] KubeVirt, "KubeVirt Architecture," [elektronski izvor]. Available: https://kubevirt.io/user-guide/architecture/. [Poslednji pristup 09 Novembar 2025].

[5] vCluster, "What is vCluster?," [elektronski izvor]. Available: https://www.vcluster.com/docs/vcluster/introduction/what-are-virtual-clusters/. [Poslednji pristup 09 Novembar 2025].

[6] The kcp Authors, "kcp Documentation," [elektronski izvor]. Available: https://docs.kcp.io/kcp/main/concepts/terminology/. [Poslednji pristup 09 Novembar 2025].

ABSTRACT

DEVELOPING OF INFRASTRUCTURE FOR LEARNING DEVOPS DEVELOPMENT PRINCIPLES

Marko Mudrinić

The adoption of *DevOps* development principles in software engineering education requires practical access to modern infrastructure, continuous integration and delivery systems, and monitoring platforms. This paper presents the design and implementation of a private cloud infrastructure developed at the School of Computer Science, University Union, Belgrade, Serbia to support learning of *DevOps* methodologies using open-source technologies.

The solution combines *Kubernetes* as the core orchestration platform with *Kubermatic* and *KubeVirt* platforms for cluster and virtualization management, enabling scalable, automated provisioning of environments for students and teaching staff. The implemented architecture provides real-world experience with infrastructure management, resource allocation, and deployment automation under limited academic resources.

The paper analyzes system performance, reliability, and maintenance aspects, highlights observed challenges such as virtualization overhead and cluster initialization latency and discusses alternative approaches using virtual *Kubernetes* clusters for further optimization and resource efficiency.