Izkušnje pri skaliranju večje rešitve z zabojniki

Andrej Krajnc, Vojko Ambrožič, Bojan Štok

IZUM – Institut informacijskih znanosti, Maribor, Slovenija
andrej.krajnc@izum.si, vojko.ambrozic@izum.si, bojan.stok@izum.si

Sistem COBISS sestavlja več aplikacij in mikrostoritev, javnost pozna predvsem spletno aplikacijo COBISS+. Pred tremi leti smo začeli prenovo aplikacije COBISS3, ki jo knjižničarji uporabljajo za poslovanje knjižnice (zaloga, nabava, izposoja, serijske publikacije, elektronski viri, izpisi). Obstoječa verzija aplikacije je večslojna in uporablja grafični uporabniški vmesnik Java Swing. Nova generacija aplikacije COBISS4 ima spletni uporabniški vmesnik, podpira vse funkcionalnosti, ki jih ima obstoječa aplikacija, pri čemer gre za kompleksno aplikacijo z nekaj 1000 poslovnimi entitetami. Obe aplikaciji uporabljata isti zaledni strežnik, ki teče za vsako od več kot 1400 knjižnic, pri čemer za večje knjižnice teče več instanc aplikacije. Nameščanje in nadgradnja več kot 2000 instanc je v virtualnem okolju zelo zahtevna, zato smo se odločili za uporabo zabojnikov. Analizirali smo dva različna orkestratorja za zabojniško okolje. Odločili smo se za uporabo Docker Swarm, ker je namestitev in vzdrževanje precej bolj enostavno kot pri orkestratorju Kubernetes. Celotni postopek smo avtomatizirali z orodjem Ansible, delovanje aplikacij in mikrostoritev pa nadziramo in spremljamo z orodjem Prometheus. Za izenačevalnik obremenitve smo uporabili Traefik. Izkušnje pri delovanju gruče so pozitivne, prav tako je sistem zelo stabilen in obvladljiv. Načrtujemo, da bomo v prihodnje uporabili zabojnike še pri razvoju drugih aplikacij v sistemu COBISS.

Ključne besede:
COBISS
COBISS Lib
zabojniki
Docker Swarm
IT arhitekture

Univerzitetna založba Univerze v Mariboru
ISBN 978-961-286-772-0
DOI https://doi.org/10.18690/um.feri.8.2023.6

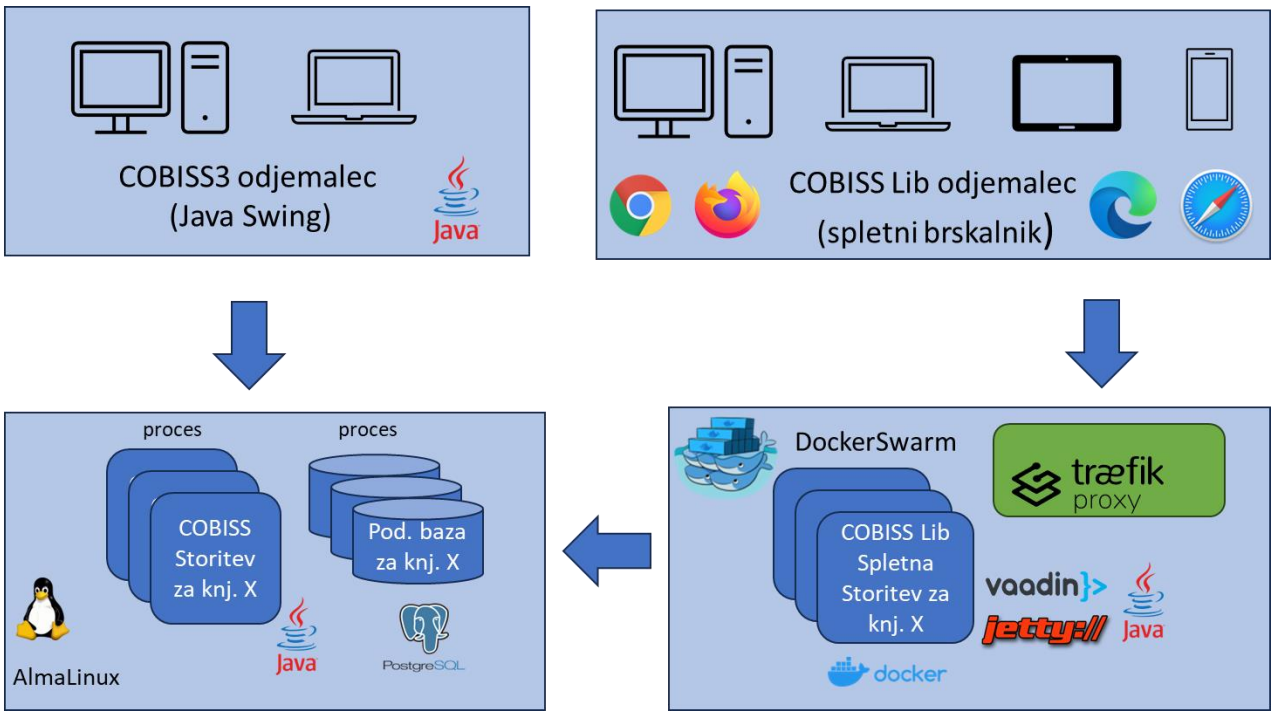
1 Uvod

Institut informacijskih znanosti (IZUM) razvija informacijski servis slovenske znanosti, kulture in izobraževanja. Javnosti najbolj poznan je sistem COBISS, ki predstavlja temelj knjižničnega informacijskega sistema Slovenije in nekaterih drugih držav, ki so povezane v mrežo COBISS.net. V mrežo COBISS.net so poleg Slovenije povezane še Srbija, Severna Makedonija, Bosna in Hercegovina, Črna gora, Albanija in Bolgarija.

Sistem COBISS ni le ena sama aplikacija, temveč zaobjema več aplikacij in mikrostoritev. Splošni javnosti sta najbolj znani spletna aplikacija COBISS+ in mobilna aplikacija mCOBISS. Ti aplikaciji uporabnikom knjižnic, ki iščejo relevantne informacije, knjižnično gradivo ali e-vire, omogočata iskanje, rezervacije, podaljšanje roka izposoje itd. Za same knjižnice je najpomembnejša knjižnična aplikacija, ki jo IZUM razvija posebej za knjižnice. COBISS3 je tretja generacija te programske opreme, ki pokrivajo posamezna ključna področja avtomatizacije knjižničnih postopkov. COBISS3 je namizna aplikacija, ki jo knjižničarji uporabljajo za poslovanje knjižnice (zaloga, nabava, izposoja, serijske publikacije, elektronski viri, izpisi). Aplikacija COBISS3 je večslojna, napisana je v programskem jeziku Java oz. s pomočjo nabora orodij za gradnike grafičnega vmesnika za Javo, torej s pomočjo knjižnice Swing.

Nova generacija knjižničnih aplikacij se imenuje COBISS Lib in je na voljo kot spletna aplikacija. Grafični vmesnik v aplikaciji COBISS Lib je narejen s pomočjo ogrodja Vaadin, ki omogoča razvoj spletnih aplikacij z uporabo spletnih komponent [1]. Aplikacija COBISS Lib podpira vse funkcionalnosti, ki jih ima obstoječa aplikacija, pri čemer gre za kompleksno aplikacijo z nekaj 1000 poslovnimi entitetami.

Obe aplikaciji (COBISS3 in COBISS Lib) uporabljata isti zaledni strežnik, ki teče za vsako od več kot 1400 knjižnic, pri čemer za večje knjižnice teče več instanc aplikacije. Nameščanje in nadgradnja več kot 2000 instanc je v virtualnem okolju zelo zahtevna, zato smo se pri COBISS Lib odločili za uporabo zabojnikov.



Slika 1: Arhitektura knjižničnih aplikacij COBISS3 in COBISS Lib.

2 Uporaba orkestratorjev za zabojniško okolje

2.1 Primerjava orkestratorjev Kubernetes in Docker Swarm

Pri uporabi zabojnikov je pomembna izbira orkestratorja za zabojniško okolje. S pomočjo orkestratorja dosežemo visoko razpoložljivost (gruče) in visoko skalabilnost storitev. Analizirali smo dva različna orkestratorja za zabojniško okolje: Kubernetes in Docker Swarm. Odločili smo se za uporabo Docker Swarm, ker je namestitev in vzdrževanje precej bolj enostavno kot pri orkestratorju Kubernetes.

V nadaljevanju je na kratko opisana primerjava orodij Docker Swarm in Kubernetes.

Swarm in Kubernetes sta orodji za orkestracijo zabojnikov, ki pomagata pri upravljanju in razporejanju storitev v oblačnih okoljih. Kubernetes je naprednejši in bolj uveljavljen, toda mi smo se odločili za Swarm, ker je brezplačen, zelo enostaven za vzpostavitev in vzdrževanje. Prav tako ni treba izvesti dodatne namestitve, saj je vključen v samo namestitev Docker. Dodatno še velja omeniti, da je dokumentacija za Swarm odlična. Ocenili smo, da kljub nekaterim pomanjkljivostim še vedno dovolj dobro ustreza našim potrebam.

Kubernetes (tudi k8s) je bolj zrel projekt z večjim in bolj razširjenim ekosistemom. Ker je del odprtokodne platforme (Cloud Native Computing Foundation), ima veliko skupnost razvijalcev in podjetij, ki prispevajo k njenemu razvoju, medtem ko je Swarm samo del ekosistema Docker.

Kubernetes ima bolj kompleksno arhitekturo. Ta temelji na skupini vozlišč, ki tvorijo gručo. Vsako vozlišče ima glavni krmilnik (ang. control plane), ki upravlja in nadzoruje delovanje gruče. Swarm ima bolj preprosto arhitekturo za upravljanje.

Kubernetes ima zmogljivejše orodje za samodejno skaliranje, ki omogoča samodejno prilagajanje števila delujočih instanc aplikacije glede na obremenitev. Swarm prav tako omogoča skaliranje, vendar nima funkcionalnosti za samodejno prilagajanje števila instanc.

Oba sistema uporabljata ukazno vrstico za upravljanje in konfiguracijo storitev, kakor tudi deklarativno konfiguracijo z uporabo datotek manifest oz. stack YAML. Te opisujejo želeno stanje sistema, ki ga sistem skuša zadostiti.

Kubernetes ima širok nabor funkcionalnosti in dodatkov, ki podpirajo bolj napredne funkcionalnosti, kot so podpora za storitve s stanjem, naprednejše upravljanje z diskovnimi kapacitetami itd.

Kubernetes je zgrajen tako, da lahko teče na kateri koli oblačni platformi, medtem ko Docker Swarm deluje samo znotraj izvajalnega okolja Docker.

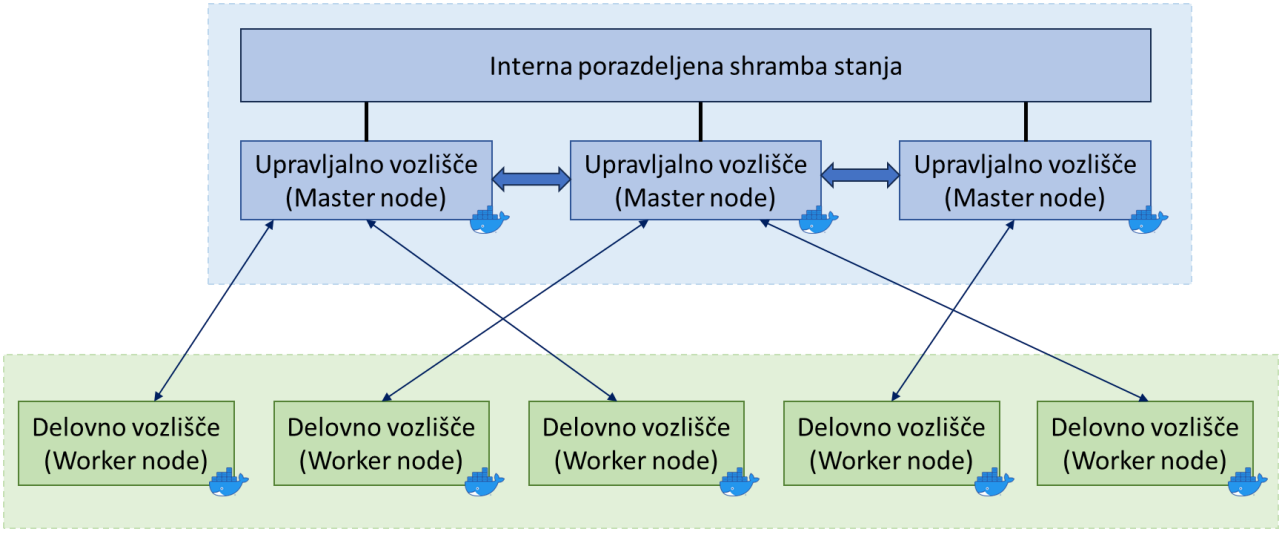
V prihodnosti bomo preverili delovanje naših storitev tudi v okolju Kubernetes. Ker pri nas uporabljamo virtualizacijo vSphere (VMware), bi bila lahko za nas zanimiva Kubernetesova rešitev "vSphere with Tanzu". To je integrirana rešitev, ki združuje virtualizacijsko platformo vSphere s Kubernetesovo platformo Tanzu. Namenjena je zagotavljanju celovitega orodja za upravljanje in delovanje zabojnikov v virtualiziranih okoljih.

2.2 Arhitektura Docker Swarm

Docker Swarm temelji na uporabi **rojev** (ang. swarm). En roj je sestavljen iz več Dockerjevih gostiteljev oz. vozlišč, ki tečejo v načinu rojev (ang. swarm mode). Ločimo dve vrsti vozlišč: upravljalna in delovna vozlišča.

Upravljalna vozlišča (ang. manager nodes) skrbijo za upravljanje roja (upravljanje Dockerjevih instanc), opravljajo funkcije orkestracije in upravljanja gruče itd. Eno izmed upravljalnih vozlišč je vodja (ang. leader) za opravljanje orkestracijskih nalog. Konsistenca stanja upravljalnih vozlišč se zagotavlja z algoritmom soglasja Raft, kjer se med kandidati izvedejo volitve in se med vozlišči določi vodja, druga vozlišča pa postanejo sledilci (ang. followers).

Delovna vozlišča (ang. worker nodes) dobivajo in izvajajo naloge (ang. task), posredovane od upravljalnih vozlišč. Privzeto tudi upravljalna vozlišča opravljajo funkcijo delovnih vozlišč, lahko pa so zgolj upravljalna vozlišča. Na vsakem delovnem vozlišču je agent, ki upravljalnim vozliščem poroča o opravilih, ki so bila dodeljena delovnemu vozlišču. Tako lahko upravljalno vozlišče vzdržuje zaželeno stanje posameznega delovnega vozlišča.

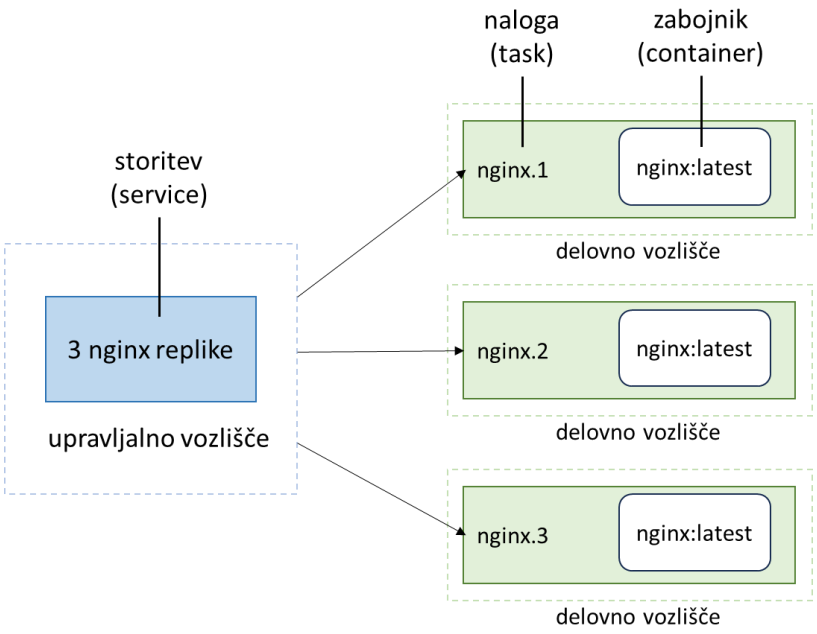


Slika 2: Docker Swarm vozlišča.

Storitev (ang. service) je definicija nalog, ki naj se izvedejo na upravljalnem ali delovnem vozlišču. Ko se storitev ustvari, se določi, katera zabojniška slika (ang. container image) naj se uporabi in kateri ukazi naj se izvedejo znotraj delujočih zabojnikov.

V primeru repliciranih storitev upravljalno vozlišče distribuira določeno število repliciranih nalog med delovna vozlišča.

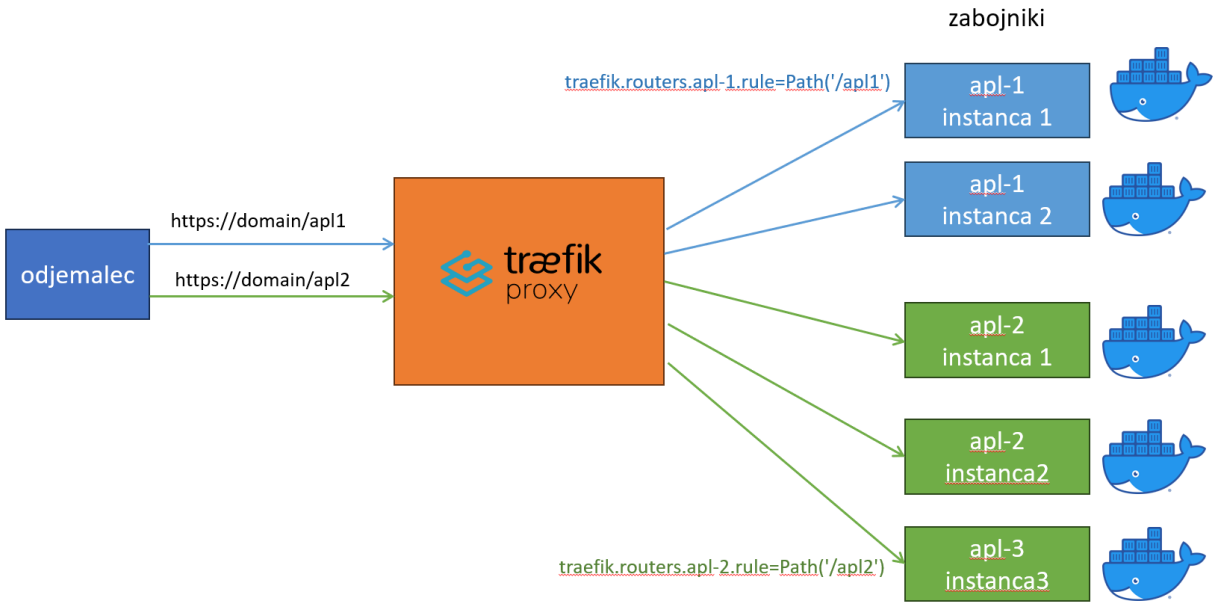
Naloga (ang. task) zaobjema Dockerjev zabojnik in vse ukaze, ki naj se izvedejo znotraj zabojnika. Naloga je atomarna enota v roju. Upravljalna vozlišča dodeljujejo naloge delovnim vozliščem glede na število replik, ki naj bi jih neka storitev imela. Ko je naloga dodeljena nekemu vozlišču, je ne moremo več prestaviti v drugo vozlišče. Naloga se na dodeljenem vozlišču izvede ali pa pade.



Slika 3: Storitve in naloge Docker Swarm.

3 Traefik

Za posrednika med zunanjimi odjemalci in našimi storitvami (dinamični reverse proxy) smo uporabili odprtokodni program Traefik, ki smo ga namestili na eno izmed upravljalnih vozlišč gruče. Traefik nadzira registracijo zabojnikov v mrežo ter nato na osnovi URL-naslova preusmeri zahteve na ustrezno instanco zabojnika naše storitve.



Slika 4: Traefik v okolju Docker.

Ker storitev hrani kontekst seje, mora Traefik preusmeriti odjemalčevo zahtevo vedno na isto instanco zabojnika (ang. sticky mode). Navodila za preslikavo zahtev dobi Traefik preko značk (ang. label) v opisu storitve (stack-datoteka).

Za visoko razpoložljivost sistema skrbi zunanji izenačevalnik obremenitve (load balancer), ki zahteve na zunanji mreži (DMZ) preusmeri na storitev Traefik na enem izmed upravljalnih vozlišč v notranjem omrežju (intranet).

4 Izgradnja in upravljanje Docker okolja

Za inicializacijo roja Swarm, dodajanje oz. odvzemanje upravljalnih in delovnih vozlišč, nameščanje Dockerjevih strežnikov in samih virtualnih strežnikov smo uporabili odprtokodno orodje Ansible, ki je namenjeno upravljanju konfiguracij, namestitvam in avtomatizaciji nalog na večjem številu strežnikov. Naloge opišemo v tako imenovanih Ansible-skriptah (Ansible Playbook).

Za upravljanje in nadzor nad Ansible-skriptami smo uporabili odprtokodno spletno platformo AWX, ki je brezplačna različica komercialne platforme Tower podjetja Red Hat. Na osnovi teh orodij je izgradnja roja Swarm rešljiva znotraj nekaj minut.

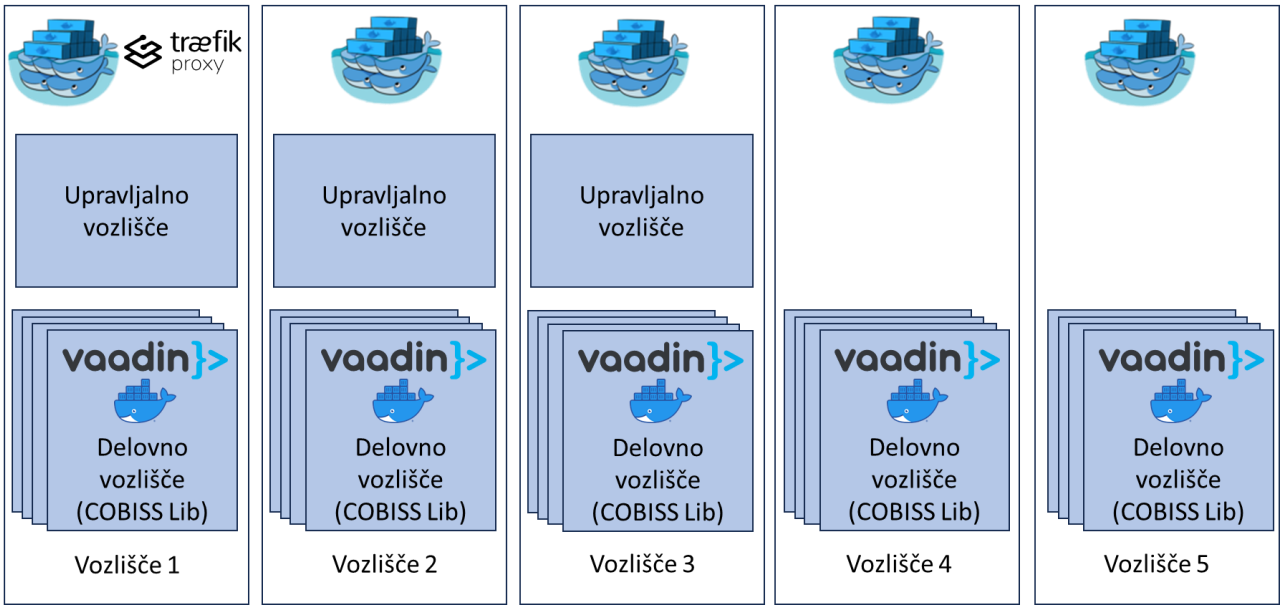
Storitve v roju Swarm so opisane v tako imenovanih stack-datotekah (format YAML). V opisu so navedeni računalniški viri, ki jih storitev potrebuje. Navedeni so parametri in razne nastavitve za storitev, vrata TCP/IP, ki jih storitev izpostavlja in marsikaj drugega. V IZUM-u imamo razvito orodje »Monitor Manager« za izgradnjo meta modela infrastrukture, strežnikov, aplikacijskih strežnikov, baz podatkov, mikrostoritev in modela knjižnic. Na osnovi tega modela se samodejno kreirajo stack-datoteke za storitve in se hkrati namestijo na upravljalna vozlišča roja. Vsaka sprememba na knjižnicah oz. infrastrukturi je takoj zaznana v storitvah roja Swarm.

```
version: '3.2'
services:
 s:
   image: dockreg.izum.pri:5000/c4client:$VERSION
   hostname: c3_nuk_dev
ports:
    - target: 8080
      protocol: tcp
      mode: host
    - target: 8787
      protocol: tcp
      mode: host
  environment:
    C4_CONF: http://dvcl011.int.izum.si/clib/par/config_nuk_dev.ini
    C4_LOG_PATH: /var/log
    C4_HOSTNAME: '{{.Node.Hostname}}'
    C4_SERVICE_NAMES: '{{.Service.Name}}'
    C4_TASK_NAME: '{{.Task.Name}}'
    C4_LOG_LEVEL: info
    TZ: Europe/Ljubljana
```

```
deploy:
   replicas: 2
   labels:
    - traefik.constraint-label=traefik-public
     - traefik.http.routers.c3_nuk_dev.rule=PathPrefix(`/clib/nuk_dev`)
     - traefik.http.services.c3_nuk_dev.loadbalancer.sticky.cookie=true
   resources:
    limits:
      cpus: '4'
      memory: 4G
    reservations:
      memory: 256M
 networks:
  c4net:
 volumes:
  - /usr/share/zoneinfo:/usr/share/zoneinfo:ro
  - /etc/localtime:/etc/localtime:ro
  - type: bind
    source: /data/log
    target: /var/log
```

Slika 5: Izsek opisa Dockerjeve storitve za COBISS Lib.

Pri razvoju programske opreme uporabljamo tudi odprtokodno orodje Jenkins za neprekinjeno integracijo (Continuous Integration – CI) in neprekinjeno dostavo (Continuous Delivery – CD). Uporablja se za avtomatizacijo gradnje, testiranja in dostave programske opreme, kar omogoča hitro in zanesljivo uvajanje sprememb v aplikaciji. V razvojnem okolju nam orodje Jenkins samodejno dvakrat dnevno generira Dockerjeve posnetke (ang. image) ter jih namesti na naš zasebni Dockerjev repozitorij in nato izvede nadgradnjo naših storitev v roju Swarm. Nadgradnja storitev v produkcijskem okolju ni samodejna in se izvede le na zahtevo.

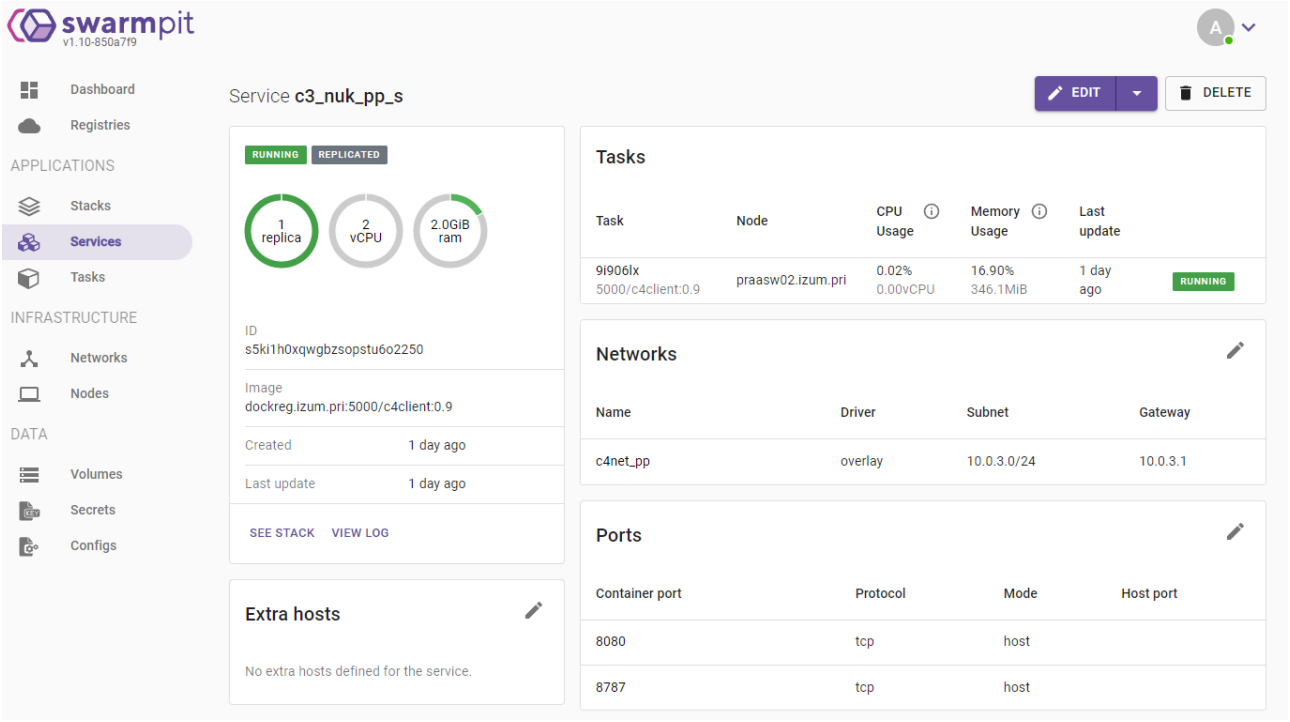


Slika 6: Okolje COBISS Lib Docker.

5 Podporna orodja v Dockerjevem okolju

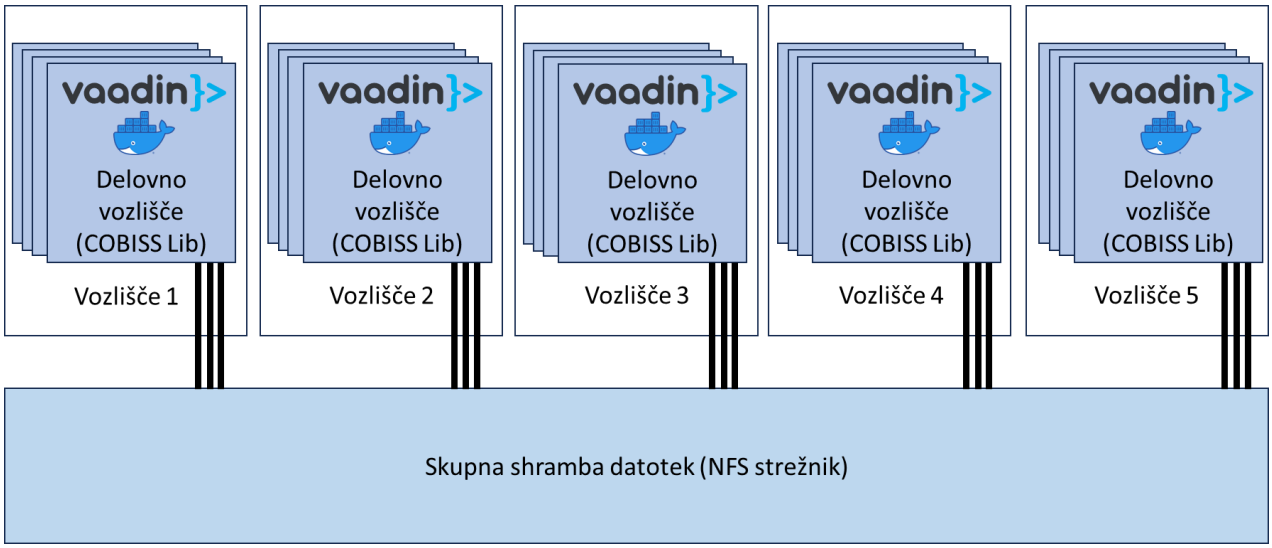
5.1 Nadzor Dockerjevih storitev

Za nadzor nad Dockerjevimi storitvami lahko uporabljamo Dockerjevo konzolo in odprtokodno platformo za upravljanje roja Swarmpit, ki je nameščena na enem izmed upravljalnih vozlišč roja.



Slika 7: Primer zaslonske slike orodja Swarmpit.

Dnevniške (ang. log) datoteke spremljamo prek konzole, zbiramo jih pa tudi na skupnem NFS-strežniku, kjer izvajamo analize teh podatkov. V prihodnosti načrtujemo namestitev vtičnika za odprtokodno orodje Elasticsearch, ki je namenjeno indeksiranju, shranjevanju in iskanju velikih količin strukturiranih in nestrukturiranih podatkov v realnem času.



Slika 8: Skupni NFS-strežnik.

Virtualne strežnike nadzorujemo tudi z odprtokodnim orodjem Icinga, ki omogoča odkrivanje težav, generiranje opozoril in spremljanje zmogljivosti.

5.2 Zbiranje in spremljanje metrik v Dockerjevem okolju

Za zbiranje in spremljanje metrik našega okolja uporabljamo odprtokodne storitve Swarmprom, ki združujejo orodja odprtokodnih projektov Prometheus in Grafana. Prometheus je sistem za spremljanje in zbiranje metrik, ki se osredotoča na zbiranje časovno vrstičnih podatkov. Spremlja zdravstveno stanje, zmogljivost in druge metrike iz različnih virov v realnem času. Metrike se shranjujejo lokalno in omogočajo analizo, vizualizacijo in generiranje opozoril prek orodja Alert manager. Grafana je platforma za vizualizacijo metrik in analizo podatkov. Omogoča ustvarjanje prilagojenih grafov, nadzornih plošč in poročil iz metrik, ki jih zbere Prometheus.

5.3 Diagnostična orodja v Dockerjevem okolju

V primeru težav z našimi storitvami lahko uporabimo bolj ali manj vsa diagnostična orodja, ki smo jih vajeni uporabljati v klasičnih okoljih. Med drugim si pomagamo z različnimi javanskimi orodji, kot so jmap, jstack itd.

6 Zaključek

Praktične izkušnje pri delovanju roja Docker Swarm so zelo pozitivne, prav tako je sam sistem zelo stabilen in obvladljiv. Čeprav je Docker Swarm manj zmogljiv orkestrator kot Kubernetes, se je izkazal kot primerna izbira, saj vsebuje vse, kar potrebujemo pri aplikaciji COBISS Lib.

Če bi se slučajno pojavile težave z obstoječo gručo, lahko zelo hitro (v nekaj minutah) namestimo novo gručo Docker Swarm, saj je sam postopek namestitve avtomatiziran.

Načrtujemo, da bomo v prihodnje uporabili zabojnike še pri razvoju novih aplikacij v sistemu COBISS in da bomo obstoječe spletne aplikacije in mikrostoritve, ki zdaj tečejo v navideznih strežnikih, preselili v zabojnike. Zabojnikov ta trenutek ne bomo uporabili za shrambo podatkovnih baz, kot so PostgreSQL, Apache Solr, ElasticSearch itd. Za te storitve bomo še naprej uporabljali virtualne strežnike.

V prihodnosti bomo sistem NFS, ki ga zdaj uporabljamo za dnevniške datoteke, nadomestili s porazdeljenimi diski Ceph in vpeljavo shrambe S3.

Literatura

[1] KRAJNC Andrej, PULKO Jani, KOROŠEC Andrej, ŠTOK Bojan, "Razvoj spletnih aplikacij z uporabo spletnih komponent", Sodobne informacijske tehnologije in storitve OTS 2022: Zbornik petindvajsete konference, Maribor, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za informatiko, 42-54.

[2] Kubernetes, https://kubernetes.io/, obiskano 13.7.2023

[3] Docker Swarm, https://docs.docker.com/engine/swarm/, obiskano 13.7.2023

[4] Traefik, https://traefik.io/traefik/, obiskano 13.7.2023

[5] Ansible, https://www.ansible.com/, obiskano 13.7.2023