Nadzor aplikacija i servisa u računalnom klasteru

Dumančić, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

NADZOR APLIKACIJA I SERVISA U RAČUNALNOM KLASTERU

Diplomski rad

Josip Dumančić

Osijek, 2018.

SADRŽAJ

1. UVOD

Nadzor računalnih mreža obuhvaća alate i procese pomoću kojih mjerimo i upravljamo računalnom mrežom kako bi smanjili opterećenje, spor odziv, potencijalne greške, povećali sigurnost, te kako bi imali uvid u računalnu mrežu i njeno stanje. Nadzor može biti aktiviran ručno, od strane korisnika ili ga uopće nema. U slučaju bilo kakve pogreške ili ne željenog odziva može se oglasiti alarm koji označava neželjeno stanje odnosno vladanje sustava. Nadzorni sustavi mogu djelovati u sklopu upravljačkih sustava tako da pokrenu predefinirane rutine za oporavak od uočene greške koje će postaviti sustav u željeno stanje ili alarmirati mrežnog administratora (putem email, SMS ili nekim drugim alarmima). Glavni zadatak nadzora računalnih mreža je smanjenje ne dostupnog vremena te efikasnije upravljanje resursima kako bi postigli što veću učinkovitost.

1.1. Zadatak diplomskoga rada

U diplomskom radu je opisan nadzor računalnog klastera i mreže pomoću Zabbix sustava. Računalni klaster sačinjavanju četiri virtualne mašine od kojih su dvije u Master načinu rada i dvije u Slave načinu rada. Klasteriranje virtualnih mašina ostvareno je pomoću Apache Mesos distribuiranog kernela. Upravljanje kontejnerima ostvaruje se pomoću Marathon radnog okvira (engl. *Framework*) za orkestriranje(upravljanje i usklađivanje) kontejnera.

2. SUSTAVI NADZORA RAČUNALNIH MREŽA

Računalne mreže su napredovale od jednodimenzionalnih mreža koje su imale znatno manje mogućnosti te su bile sačinjene od manje elemenata nego što to imaju današnje mreže [11]. Razvojem postojećih tehnologija i pojavom novih tehnologija kao što su računalni oblak (engl. *Cloud*), bežični i udaljeni pristup (engl. *Wireless, Remote access*), virtualne privatne mreže (engl. *Virtual private networks*), interneta stvari (engl. *Internet of Things*), mobilnih uređaja itd., računalne mreže su napredovale i postale sveprisutne u današnjim sustavima, gdje postoji velika potreba za mrežnim nadzorom [1].

Nadzor računalnih mreža omogućuje mrežnim administratorima uvid u stanje mreže, ispravnost povezanosti (engl. *Connection*) i povezivosti (engl. *Connectivity*), stanje na raznim mrežnim elementima, stanje na mrežnim čvorovima (engl. *Network Node*), raspoloživost, distributivnost, stanje vatrozida, informacije o komponentama koje korisnik koristi, itd. Kako bi uspješno nadzirali računalnu mrežu ili poslužitelj potrebna je dostupnost sljedećim informacijama:

- Informacije o svrsi, zdravlju, trenutnom stanju i performansama mrežnih elemenata koje nadziremo.
- Aplikacija ili software za nadzor koji moraju imati mogućnost prikupiti, obraditi i prezentirati podatke u obliku koji je pogodan za korisnika. Programska podrška (engl. *Software*) podržava mogućnost alarmiranja korisnika o potencijalnim problemima koji ovise o razini koju korisnik postavi za određeni tip alarma.
- Protokol ili metoda za prijenos informacija od elementa koji nadziremo do software-a za nadzor.
- Odgovarajući mrežna vrata (engl. *Network Port*) pomoću kojih se vrši komunikacija pomoću protokola između nadziranog elementa i nadzornog sustava[11].

Informacije koje se prikupe na mreži pomažu kod upravljanja i kontrole nad mrežom, kod identifikacije potencijalnih problema prije nego dođe do prekida rada mreže, te kod bržeg otklanjanja kvara nakon što dođe do prekida. Konstantni nadzor nam omogućava mreže visokih performansi.

Neke od metoda koje se koriste kod nazora računalnih mreža su:

- Ping

  Koristi se za testiranje dostupnosti udaljenih hostova unutar mreže. Alat ping šalje ICMP (Internet Control Message Protocol) zahtjeve za odgovorom od udaljenog poslužitelja ili

mrežnog sučelja. Odziv ping-a nam pokazuje da li je host aktivan te vrijeme potrebno za slanje i primanje podataka.

- SNMP (engl. *Simple Network Management Protocol*)

  SNMP je protokol za nadzor računalnih mreža koji se koristi za razmjenu informacija između hosta i mreže, koji je ujedno i najrasprostranjeniji protokol za upravljanje i nadzor računalnih mreža. SNMP agent dohvaća zahtjeve na portu 161, a agent šalje odgovor na zahtjev preko porta 162. SNMP protokol je dio aplikacijskog sloja OSI-ja (Open Systems Interconnection model). Protokol uključuje komponente kao što su:

- Upravljani uređaji: Mrežni čvor u mreži koji podržava SNMP i ima pristup određenim informacijama na tom čvoru.
- Agent: Software koji je dio uređaja koji nadziremo. Agent ima pristup MIB-u (Management information database) uređaja, te dopušta NMS (Network Management System) sustavu čitanje i pisanje unutar MIB-a
- Network Management System: Programska podrška koja nadzire i kontrolira upravljane uređaje putem SNMP protokola.

  Podaci sa mrežnih elemenata se prikupljaju pomoću SNMP protokola bilo da se koristi povlačenje (engl. *Polling*) podataka što čini aktivni način rada ili se koriste Trap-ovi. Trapovi čine pasivan način rada te omogućuju periodično slanje informacija NMS-u o događajima na uređaju putem SNMP agenta.

  MIB sadrži informacije o strukturi podataka na uređaju koji nadziremo. Također sadrži OID(object identifier) koji je identificira varijable kako bi se mogle pročitati i modificirati na uređaju.

- Syslog

  Dio operacijskog sustava koji se koristi za pohranu logova te dopušta uređaju slanje obavijesti događaja unutar računalne mreže. Informacije iz logova se mogu koristiti za upravljanje sustavom, kao i za povećanje sigurnosti. Syslog je podržan na raznim uređajima, od printera do routera i vatrozida.

- Bash skripte

  U računalnim mrežama gdje je NMS ne dostupan za nadzor ili postojeći NMS ne podržava posebne funkcije koristimo bash skripte. Skripte koriste standardne naredbe, te

mogu između ostaloga raditi promjene na konfiguraciji uređaja ili postaviti zadatak koji će se izvršiti u zadano vrijeme [1][11].

Primjer produkcijskog klastera na kojem se vrši nadzor, a sastoji se od 926 čvorova te može nadzirati preko 5 milijuna korisnika prikazan je na slici 2.1.

Slika 2.1. Slika ekrana prikazuje Zabbix produkcijski klaster.

2.1. Zabbix sustav za nadzor

Zabbix je programska podrška koja vrši nadzor više parametara računalne mreže, te provjerava zdravlje i integritet poslužitelja. Zabbix koristi fleksibilne mehanizme obavještavanja koji omogućuju korisniku konfiguriranje alarma za bilo koji događaj, koji se javljaju mrežnom administratoru. To nam omogućuje brzu reakciju na potencijalne probleme s poslužiteljem. Zabbix nam također omogućuje izvještaje i vizualni prikaz spremljenih podataka. Zbog toga se Zabbix koristi za planiranje potrebnog kapaciteta.

Podržav dva načina nadzora, aktivni i pasivni. Grafovi, mape, izvještaji, statistike, te konfiguracijski parametri su dostupni putem Zabbix WebUI (eng. Web user interface-a). Web UI nam omogućuje prikaz stanja i zdravlja računalne mreže s bilo koje lokacije. Pravilno konfigurirani Zabbix igra veliku ulogu u nadzoru računalnih mreža, kako za male organizacije koje imaju nekoliko poslužitelja, tako i za velike organizacije koje imaju više stotina poslužitelja.

Zabbix je napisan i distribuiran pod GPL (General Public License) version 2., što znači da je distribuiran i dostupan za općenitu uporabu besplatno. Zabbix je visoko integrirano programsko rješenje za nadzor računalnih mreža, a neke od mogućnosti Zabbix-a su:

- Prikupljanje podataka - provjera dostupnosti i svojstva, podrška za SNMP, IPMI (eng. Intelligent Platform Management Interface) , JMX (eng. Java Management Extensions) nadzor, prikupljanje željenih informacija u određenom vremenskom intervalu).

- fleksibilno određivanje uvjeta na koje se oglašava alarm – korisnik određuje uvijete za koje se oglašava alarm .
- vizualni prikaz u stvarnom vremenu – prikaz prikupljenih podataka u obliku koji je pogodan za analizu.
- nadzor putem WebUI – pristup informacijama o stanju mreže s udaljene lokacije.
- povijest korištenja podataka – podatci se spremaju u bazu podataka koja ima ugrađene housekeeping procedure.
- jednostavna konfiguracija
- korištenje predložaka – za upite koji se ponavljaju odnosno odgovaraju određenoj grupi mrežnih elemenata ili određenoj funkcionalnosti.
- sigurnu autentifikaciju korisnika – korisnicima možemo definirati dijelove koji su im dostupni [2].

2.2. Arhitektura Zabbix sustava za nadzor

Zabbix se sastoji od nekoliko glavnih komponenti, a to su:

- Zabbix poslužitelj (Zabbix server)
- Baza podataka (Database)
- Zabbix WebUI (Zabbix Frontend interface)
- Zabbix Proxy
- Zabbix Agent

Da bi se razumjela arhitektura Zabbix sustava za nadzor i način rada važno je znati tok podataka unutar Zabbixa. Kako bismo npr. dobili obavijest kada opterećenje CPU-a pređe zadanu vrijednost na poslužitelju X, potrebno je stvoriti host-a za poslužitelj X. Nakon stvaranja hosta potrebno je stvoriti predmet za nadzor stanja CPU-a te postaviti okidač koji se aktivira nakon što opterećenje CPU-a pređe zadanu vrijednost. Sljedeći korak je stvaranje akcije koja šalje obavijest mrežnom administratoru. Izvedba procedure je olakšana izradom predložaka. Zabbix se koristi u područjima kao što su: IT i telekomunikacije, zdravstvo, bankarstvo, edukacija, marketing, zrakoplovna industrija, itd [2].

2.2.1. Zabbix poslužitelj

Zabbix poslužitelj je centralna komponenta Zabbix programskog rješenja, prema kojemu Zabbix agent i Zabbix proxy šalju podatke o stanju i integritetu sustava. Poslužitelj obavlja aktivnu i pasivnu provjeru podataka, te šalje obavijesti korisniku. Poslužitelj može sam provjeriti stanje mrežnih servisa (kao što su Web i Mail poslužitelji) koristeći jednostavne ugrađene provjere. Na poslužitelju se nalaze sve konfiguracije domaćina odnosno mrežnih elemenata kao i statistike te prikupljeni podatci. Na osnovu definiranih pravila i prikupljenih podataka sustav može stvoriti alarm koji se prosljeđuje administratorima na odgovarajući način. Zabbix poslužitelj čine tri komponente koje mogu biti na istom domaćinu ili na zasebnim domaćinima u odgovarajuću povezivost između komponenti:

- Zabbix poslužitelj (Zabbix Server)
- Web sučelje (Zabbix Web Frontend)
- Baza podataka (Zabbix Database)

Sve konfiguracijske postavke su pohranjene u bazi podataka, kojoj mogu pristupiti poslužitelju i web frontend. Prilikom stvaranja nove stavka (engl. *Item*) putem web frontend (ili aplikacijskog programskog sučelja, API), stavka se dodaje u bazu podataka gdje sprema vrijednost prikupljenih podataka u odgovarajuće tablice. Korisnik ima mogućnost definiranja vremenskog intervala prilikom kojeg Zabbix poslužitelj ciklički provjerava sve aktivne stavke te prikupljene informacije pohrani u priručnu memoriju (engl. *Cache*) odnosno zapisuje u bazu podataka. Prethodno opisani slijed može uzrokovati sporijim odzivom sustava na unesene promjene putem Zabbix web sučelja. Promjene u konfiguraciji su vidljive u sustavu sa zakašnjenjem koje može ovisiti o nekoliko faktora kao što je mrežne latencije, vrijeme između ponovnog učitavanja priručne memorije, itd.

Pokretanje poslužitelja se postiže dolje navedenom naredbom:

```
shell> ./zabbix_server
```

Moguće je proslijediti sljedeće parametre sa Zabbix poslužitelju:

```
-c --config <file>              apsolutna putanja do konfiguracijske datoteke (ako
nije promijenjena /usr/local/etc/zabbix_server.conf)
-n --new-nodeid <nodeid>        pretvara bazu podataka u novi nodeid
-R --runtime-control <option>   administracijske funkcije
-h --help                       upute
```

```
-V --version                    prikaz trenutne verzije
```

Za pokretanje Zabbix poslužitelja potrebno je definirati zabbix korisnika sa odgovarajućim pravima izvršavanja. Zabbix korisnik dolazi sa instalacijom zabbix poslužitelja na domaćinu. Pokretanje zabbix posužitelja pomoću „root“ korisnika je moguće uz prethodno podešenu opciju „AllowRoot“ unutar konfiguracijske datoteke zabbix poslužitelja. U protivnom, pokretanje zabbix poslužitelja sa „root“ korisnikom će automatski prebaciti na predprogramiranog korisnika „zabbix“. Ako se poslužitelj i Agent pokreću s istog računala preporučljivo je koristiti različitog korisnika za pokretanje poslužitelja i Agenta, jer ako se pokrenu s istim korisnikom, Agent ima pristup konfiguraciji poslužitelja te bi mogao jednostavno doći do zaporke od baze podataka [2].

Zabbix poslužitelj je podržan na sljedećim platformama:

- Linux
- Solaris
- Mac OS X
- FreeBSD
- OpenBSD
- SCO Open Server
- Tru64/OSF1
- AIX
- HP-UX

2.2.2. Zabbix Proxy

Zabbix proxy je komponenta koji nadzire i prikuplja podatke s jednog ili više uređaja, svi podatci su spremljeni lokalno, te se šalju Zabbix poslužitelju kojemu proxy pripada. Integracija proxy-a nije obavezna, ali je jako korisna za raspodjelu opterećenja u sustavu. Ako samo proxy prikuplja podatke, poslužitelji zahtijevaju manje CPU-a te manji memorijski prostor jer se prikupljanje i obrada podataka vrši na proxy-u, te se ti podatci prosljede poslužitelju na daljnu obradu. Integracija proxy-a je idealno rješenje za centralizirani nadzor udaljenih lokacija, grana i mreža koje nemaju svog lokalnog administratora. Zabbix proxy zahtjeva svoju odvojenu bazu podataka.

Pokretanje Zabbix proxy-a se postiže unosom sljedeće naredbe:

```
shell> service zabbix-proxy start
```

Naredbu je moguće pokrenuti na većini GNU/Linux sustava, dok je na ostalim operacijskim sustavima to moguće izvršenjem sljedeće naredbe:

```
shell> /etc/init.d/zabbix-proxy start
```

Također je moguće zaustaviti/ponovno pokrenuti/dobiti status sljedećim naredbama:

```
shell> service zabbix-proxy stop
shell> service zabbix-proxy restart
shell> service zabbix-proxy status
```

Moguće je proslijediti sljedeće parametre sa Zabbix proxy-em:

```
-c --config <file>              apsolutna putanja do konfiguracijske datoteke
-R --runtime-control <option>   administracijske funkcije
-h --help                       upute
-V --version                    prikaz trenutne verzije
```

Za pokretanje Zabbix proxy-a nije potreban „root“ korisnik, te ako ga pokrenemo kao „root“, automatski će se prebaciti na predprogramiranog korisnika „zabbix“ koji mora biti instaliran na sustav na kojem se pokreće. Moguće je pokrenuti proxy kao „root“ jedino ako promijenimo „AllowRoot“ parametar unutar konfiguracije proxy-a. Zabbix proxy je moguće pokrenuti na istim operacijskim sustavima kao i Zabbix poslužitelj. Proxy zahtjeva UTF-8 lokalno kako bi mogao točno interpretirati podatke. Većina Unix sustava automatski podržava UTF-8, dok se na ostalim sustavima treba posebno definirati[2].

2.2.3. Zabbix Agent

Zabbix agent se instalira na domaćinu kojem želimo nadzirati resurse i aplikacije (stanje memorije, statistiku procesora, promet, itd.). Agent prikuplja podatke te ih prosljeđuje poslužitelju, ili proxy-u, na daljnju obradu. U slučaju kvara (npr. ako je hard disk popunjen ili je neki od procesa postao neaktivan), Zabbix poslužitelj može automatski obavijestiti administratore o kvaru. Nadziranje sustava pomoću zabbix agenta je efikasano jer se koriste sustavski pozivi za prikupljanje podataka o stanju.

Postoje dva načina provjere:

- Pasivna provjera – agent odgovara za zadani upit od strane Zabbix poslužitelja odnosno zabbix proxy-ja. Agent prihvaća dobiveni upit, izvršava odgovarajuće naredbe na sustavu, te prosljeđuje dobivene informacije zabbix poslužitelju odnosno zabbix proxy-ju.
- Aktivna provjera – agent prvo pokupi listu predmeta za nadzor od Zabbix poslužitelja. Zatim periodički šalje novu vrijednost poslužitelju odnosno proxy-ju.

Kod pasivnog i aktivnog nadzora potrebno je odabrati vrstu predmeta koji nadziremo. Zabbix agent procesuira predmete tipa „Zabbix agent“ i „Zabbix agent (active)“. Podržan je na jednakim operacijskim sustavima kao Zabbix poslužitelj i Zabbix proxy [2].

Pokretanje agenta nakon instalacije moguće je sljedećim unosom:

```
shell> service zabbix-agent start
```

Također je moguće zaustaviti/ponovno pokrenuti/dobiti status agenta sljedećim naredbama:

```
shell> service zabbix-agent stop
shell> service zabbix-agent restart
shell> service zabbix-agent status
```

Zabbix agent vraća return 0 u slučaju da je zadatak uspješno obavljen i 1-255 u slučaju greške[2].

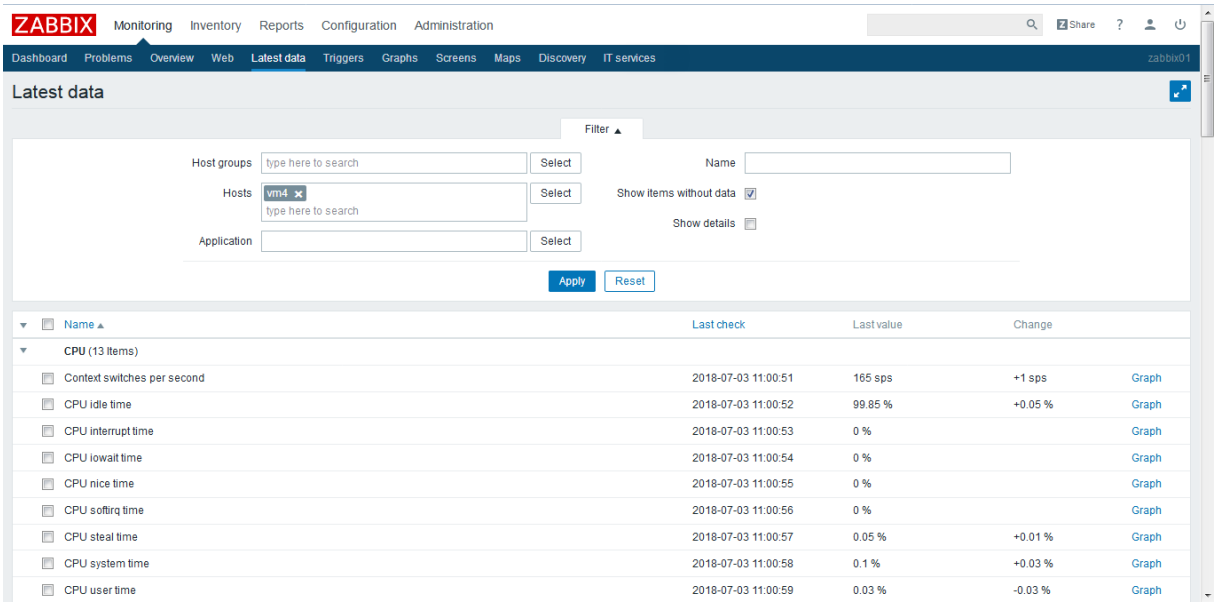
2.2.4. Zabbix mrežno korisničko sučelje

Zabbix web frontend je komponenta Zabbix sustava koja ga odvaja od konkurencije. Jednostavno za korištenje, ali moćno grafičko sučelje kojim možemo nadzirati računalne mreže. Zbog jednostavnosti i mogućnosti konfiguracije pogodan je za nove korisnike i administratore sustava.

Mogućnosti Zabbix WebUI-a su:

- Grafički prikaz- mogućnost prikaza više grafičkih dijagrama na jednom ekranu, te osvježava informacija u zadanom periodu.
- Jednostavna konfiguracija – mogućnost promjene konfiguracije putem web sučelja, bez potrebe za znanjem Unix uređivača teksta (engl. *Text editor*) i konzole.

- Kontrola na jednom mjestu – olakšava nadzor i prikupljanje podataka o stanju sustava unutar jednog sučelja.
- Nije potreban restart – centralizirana konfiguracija omogućava izmjene bez potrebe za restartom poslužitelja.
- UTF-8 – podrška Unicode/UTF-8.
- Pregled prijašnjih stanja – sve operacije se pohranjuju u logove.
- Pretraga – pretraga unosom traženog stringa. String se uspoređuje s hostovima, nije osjetljivo na velika i mala slova.
- Više jezičnost- preveden u potpunosti na nekoliko jezika i taj broj se povećava.
- Obavijesti – prikaz obavijesti koje zahtijevaju hitnu reakciju.
- Mapiranje – mogućnost prikaza okolina koje nadziremo na željenoj pozadini, radi lakšeg prikaza strukture nadzirane mreže korisniku.
- Prikaz prometa – prikaz podatkovnog prometa kroz poslužitelj u stvarnom vremenu.
- Status – prikaz stanja sustava i njegovih pojedinih komponenti.
- Prezentacija – olakšava prezentaciju stanja sustava korisniku [3].



Slika 2.2. Slika ekrana prikazuje Zabbix mrežno korisničko sučelje.

3. RAČUNALNI KLASTERI I NJIHOVA PRIMJENA

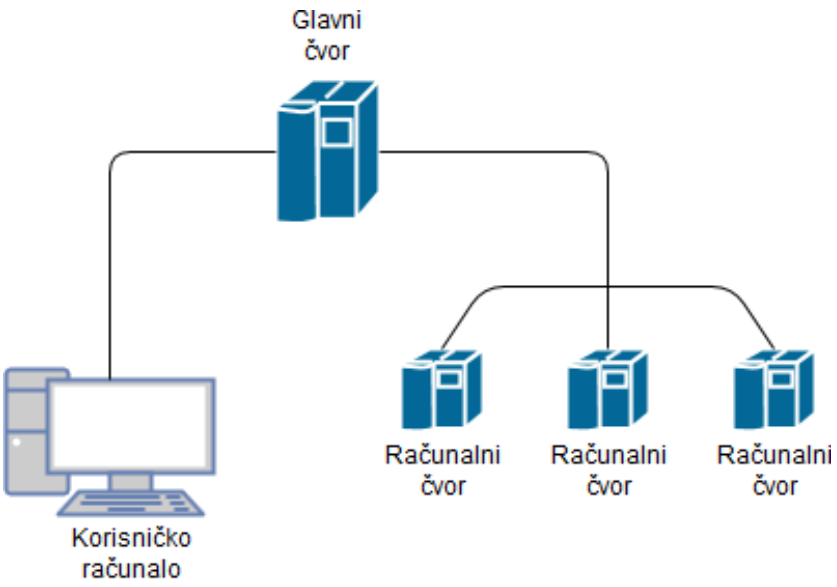
Računalni klaster je logička jedinica koja se sastoji od više računala povezanih putem LAN (engl. *Local Area Network*). Povezana računala rade kao jedna cjelina. Računalni klasteri nam omogućuju puno veću brzinu obrade, više memorije za pohranu podataka, veći integritet podataka, veću pouzdanost te veću dostupnost resursa. Teži su za implementaciju i održavanje, te su skuplji, ali nam omogućuju veće performanse [4].

Izvedbe klastera su:

- Klasteri za balansiranje opterećenja (engl. *Load-balancing*)
- Klasteri velike dostupnost (engl. *High availability (HA)*)
- Klasteri velikih performansi (engl. *High performance(HP)*)

Glavne prednosti korištenja računalnih klastera vidimo kad trebamo obraditi veliku količinu podataka. Kad se koristi za navedene svrhe računalni klasteri nam omogućuju:

- Isplativost – isplativo s obzirom na količinu energije i brzinu obrade koju postižemo. Efikasnije i jeftinije u odnosu na druga rješenja, kao što su superračunala.
- Brzina obrade – više računala velike brzine rade zajedno kako bi omogućili ujedinjeno obrađivanje što je ujedno i brže.
- Poboljšana mrežna struktura – drugačije LAN topologije za implementaciju računalnog klastera, koje sprječavaju pojavu zagušenja.
- Fleksibilnost – za razliku od superračunala, računalni klasteri se mogu nadograditi te tako poboljšati postojeće specifikacije ili dodati dodatnu komponentu u sustav.
- Visoka dostupnost resursa – ako dođe do greške na jednoj od komponenti unutar računalnog klastera, ostali dijelovi nastavljaju s radom, što je velika razlika u odnosu na superračunala [4].



Slika 3.1. Računalni klaster.

3.1. Apache Mesos

Apache Mesos je upravitelj klasterima baziran na otvorenom kodu koji upravlja radnim opterećenjem u distribuiranom okruženju, dijeljenjem resursa i izolacijom. Mesos je pogodan za implementaciju i upravljanje aplikacijama u velikim klasteriranim okruženjima. Mesos spaja postojeće resurse računala / čvorova u klasteru u jedan bazen koji se poslije mogu koristiti za obavljanje različitih poslova. To se još naziva "apstrakcija nodova", što nam uklanja potrebu za dodjeljivanjem računala za različita radna opterećenja. Korporacije kao što su Twitter, Airbnb i mnogi drugi koriste Apache Mesos. Mesos se nalazi između operacijskog sustava i aplikacijskog sloja te djeluje kao respodjeljeni kernel (engl. *Distributed Kernel*) podatkovnog centra. Mesos izolira procese koji se izvode u klasteru, kao što su memorija, procesor, datotečni sustav i I / O, kako se ne bi međusobno ometali. Takva izolacija omogućuje Mesosu stvaranje jedinstvenog, velikog bazena resursa koji obavlja radne zadatke. Apache Mesos koristi arhitekturu master-agent u kombinaciji s okvirima za upravljanje i izoliranje zahtjeva za resursima. Agent deamon je aktivan na svakom čvoru unutar klastera. Postoji više okvira od kojih svaki ima posebnu ulogu. Marathon je aplikacijski okvir koji je usmjeren na upravljanje servisima koji se dugo izvršavaju, te za specijalizirane skalabilne servise, što je pogodno za kontinuirani nadzor. Ako se ne koriste okviri određena opterećenja mogu potrošiti sve dostupne resurse. Mesos također koristi Apache Zookeeper, dio Hadoopa, pomoću kojeg sinkronizira distribuirane procese kako bi svim klijentima osigurali dosljedne podatke i osigurali toleranciju kvarova. Svaki od radnih

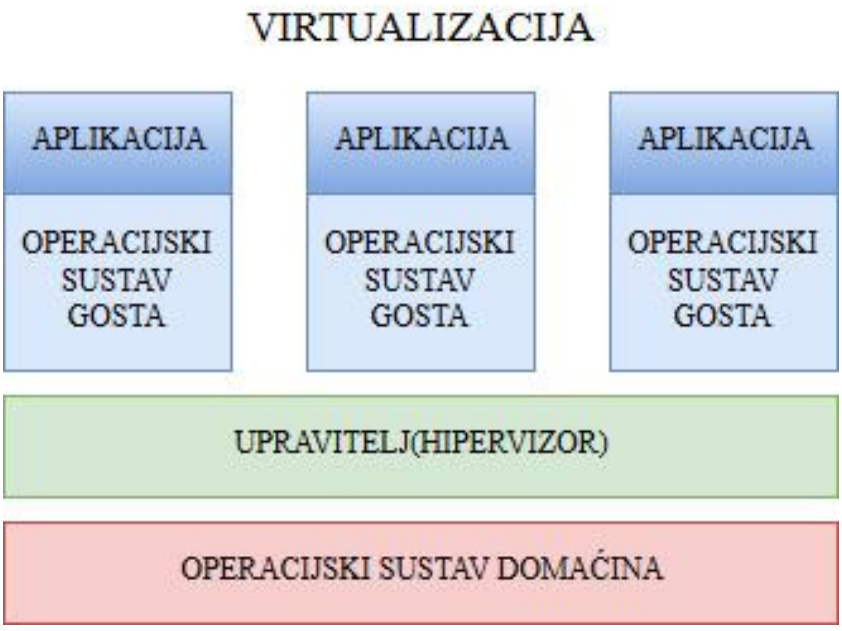
okvira sastoji se od dvije ključne komponente: planer i izvršitelj. Planeri se povežu s Mesos upraviteljem (engl. *Mesos Master*) kako bi dobili resurse, a izvršitelji pokreću naredbu ili program koji izvršava zadatke na slave-u. Master dodjeljuje resurse svakom okviru, ali planer odabire koji od tih raspoloživih resursa treba koristiti. Nakon što okvir dobije potrebne resurse, šalje natrag opis zadataka master-u. Zatim master šalje zadatke slave-u, te izvršitelj na slave-u pokrene zadatak. Mesos može upravljati Docker kontejnerima pomoću Chronos i Marathon okvira. Mesos također daje mogućnost korištenja planiranja više resursa [5].

3.2. Apache Zookeper

Apache Zookeper je alat otvorenog koda koji omogućuje centraliziranu infrastrukturu i usluge koje omogućuju sinkronizaciju putem Apache Hadoop klastera. ZooKeeper održava zajedničke objekte potrebne u velikim klasteriziranim okruženjima. Primjeri tih objekata uključuju informacije o konfiguraciji, hijerarhijski prostor za imenovanje i tako dalje, koje aplikacije koriste za koordiniranje distribuirane obrade unutar velikih klastera. Za aplikacije, ZooKeeper pruža infrastrukturu za sinkronizaciju unakrsnih čvorova. Sinkronizacija se postiže održavanjem informacija o statusu u memoriji na ZooKeeper poslužiteljima. ZooKeeper poslužitelj čuva kopiju stanja cijelog sustava i pohranjuje te podatke u lokalnim logovima. Veliki klasteri sadrže više ZooKeeper poslužitelja (glavni poslužitelj sinkronizira poslužitelje naviše razine). Na ZooKeeper poslužitelju, aplikacija može stvoriti takozvani znode (datoteka koja se pohranjuje u memoriju ZooKeeper poslužitelja). Znode se može ažurirati bilo kojim čvorom u klasteru, a svaki se čvor u klasteru može registrirati kako bi bio obaviješten o promjenama u tom znode (poslužitelj može biti postavljen tako da nadzire određeni znode). Koristeći ovu znode infrastrukturu, aplikacije mogu sinkronizirati svoje zadatke diljem distribuiranog klastera ažuriranjem njihovog statusa u ZooKeeper znode-u, što dalje može obavijestiti ostatak klastera o promjeni statusa određenog čvora. Ova usluga centralizacije statusa na razini klastera je neophodna za poslove upravljanja i serijalizacije u velikim distribuiranim poslužiteljima [6].

4. VIRTUALIZACIJA OPERACIJSKOG SUSTAVA

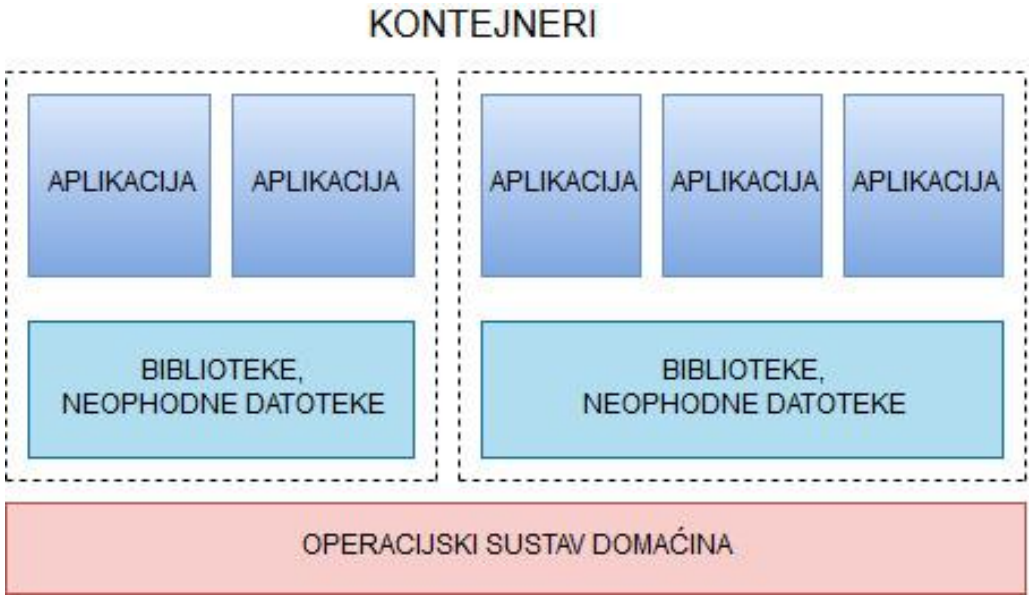
Virtualizacijska tehnologija omogućava lakše prilagođavanje standardnih operacijskih sustava kako bi se mogle pokrenuti različite aplikacije koje obrađuju više korisnika na jednom računalu istovremeno. Operacijski sustavi se ne ometaju međusobno, iako su na istom računalu. U OS virtualizaciji operacijski je sustav promijenjen tako da funkcionira kao nekoliko različitih, pojedinačnih sustava. Virtualizirano okruženje prihvaća naredbe različitih korisnika koji izvode različite aplikacije na istom računalu. Korisnici i njihovi zahtjevi su upravljani virtualiziranim operativnim sustavom. Virtualizacija operacijskog sustava omogućava korisnicima transparentnu virtualizaciju razdvajanjem aplikacija od operacijskog sustava. Tehnologija virtualizacije OS nudi veću kontrolu na razini aplikacije olakšavajući transparentnu migraciju pojedinačnih aplikacija. Virtualizacija nudi veću fleksibilnost, što rezultira smanjenim troškovima. Virtualizacija OS-a također se može koristiti za premještanje kritičnih aplikacija na drugu instancu operacijskog sustava. Zakrpe i ažuriranja na osnovni operativni sustav obavljaju se pravodobno i imaju mali ili nikakav utjecaj na dostupnost aplikacijskih usluga. Procesi u virtualiziranom okruženju OS-a su izolirani te se nadzire njihova interakcija s primarnom instancom operacijskog sustava [7].



Slika 4.1. Struktura virtualiziranog operacijskog sustava.

4.1. Docker

Docker je open source alat za automatizaciju implementacije aplikacija kao prijenosnih, samodostatnih kontejnera koji se mogu izvoditi na oblaku ili na lokalnoj razini. Kontejneri omogućuju razvojnom programeru pohranu aplikacija sa svim potrebnim dijelovima, kao što su biblioteke i druge zavisne datoteke te ih isporučuje kao jedan paket. Tako, zahvaljujući kontejneru, programer može biti siguran da će se aplikacija pokrenuti na bilo kojem drugom operacijskom sustavu bez obzira na prilagođene postavke koje računalo može imati koji bi se mogli razlikovati od uređaja koji se koristi za pisanje i testiranje koda. Na neki način, Docker je pomalo poput virtualnog računala. No, za razliku od virtualnog računala, umjesto stvaranja cijelog virtualnog operacijskog sustava, Docker omogućuje aplikacijama korištenje istog Linux kernela kao i sustav na kojemu se pokreće i zahtijeva samo slanje aplikacija sa stvarima koje već nisu pokrenute na glavnom računalu. To daje značajan napredak u izvedbi i smanjuje veličinu aplikacije. Docker donosi sigurnost aplikacijama u zajedničkom okruženju, ali kontejneri sami po sebi nisu alternativa poduzimanju odgovarajućih sigurnosnih mjera [8][9].



Slika 4.2. Struktura Docker kontejnera.

4.2. Platforma za orkestriranje i upravljanje kontejnerima

Proces implementacije više kontejnera za izvršavanje aplikacija može se optimizirati automatizacijom. To postaje sve više vrijedno jer broj kontejnera i hostova raste. Ovakav tip automatizacije naziva se orkestracija. Orkestracija može uključivati niz mogućnosti, uključujući:

- Dostupnost hostova
- Instalacija skupa kontejnera
- Premještanje neuspjelih kontejnera
- Povezivanje kontejnera putem dogovorenog sučelja
- Dopuštanje pristupa uslugama uređajima izvan klastera
- Skaliranje klastera dodajući ili uklanjajući kontejnere

Postoji više alata za orkestraciju dostupnih za Docker. Mi ćemo koristiti Marathon.

4.2.1. Marathon

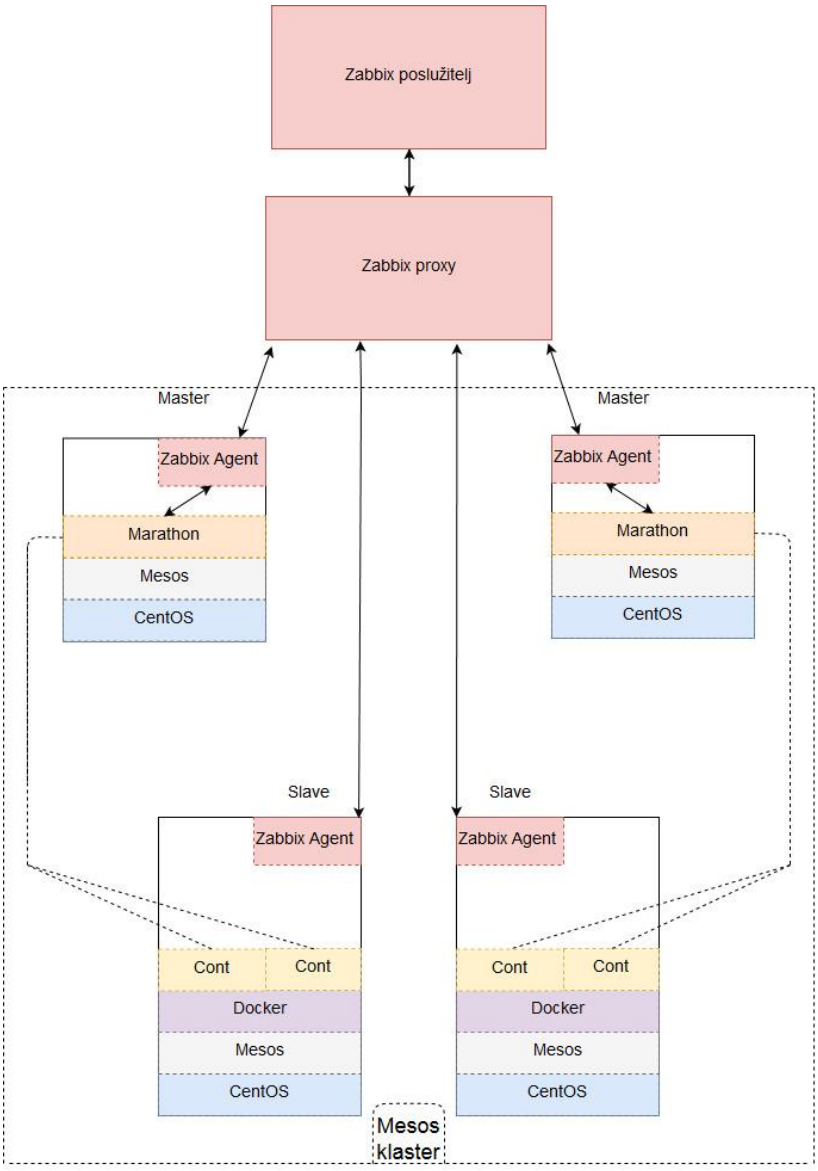
Marathon je platforma za orkestraciju kontejnera za DCOS (engl. *Datacenter Operating System*) i Apache Mesos-a.

Neke od mogućnosti Marathona su:

- Visoka dostupnost – Marathon radi kao aktivno/pasivni klaster s izborom vođe za 100% vremena.
- Pokretanje više kontejnera - Marathon ima prvorazrednu podršku za Mesos kontejnere (koristeći cgroups) i Docker.
- Korisničko sučelje
- Stateful aplikacije - Maraton može vezati trajne veličinu prostora za pohranu aplikacija. Možete pokrenuti baze podataka kao što su MySQL i MariaDB, te imati proctor za pohranu za Mesos.
- Ograničenja - omogućuju npr. postavljanje samo jednog primjera aplikacije po čvoru.
- Otkrivanje usluge i balansiranje opterećenja - Nekoliko dostupnih metoda.
- Provjera zdravlja - Procjena zdravlja programa pomoću HTTP ili TCP provjera [10].

5. TEHNIČKI OPIS PROJEKTA

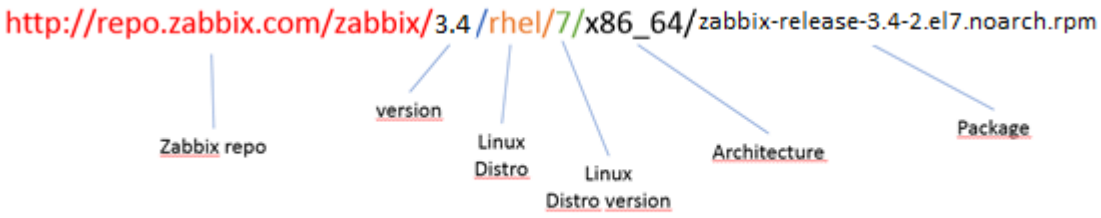
Cilj projekta je stvoriti i konfigurirati računalni klaster unutar kojega ćemo nadzirati stanje u sustavu. Struktura projekta se sastoji od Zabbix poslužitelja, Zabbix proxy-a te Mesos klastera. Mesos klaster sačinjavaju četiri računala od kojih su, dva u master, te dva u slave načinu rada. Zabbix poslužitelj prikuplja podatke s master čvorova u klasteru pomoću Zabbix proxy-ja. Na master čvorove postavljen je Marathon koji omogućava integraciju i upravljanje dockeriziranim aplikacijama. MySQL baza podataka i HTTP poslužitelj se nalaze unutar docker kontejnera na slave čvorovima unutar klastera. Navedene dockerizirane aplikacije se nadziru pomoću Zabbix sustava za nadzor uz dodatak bash skripti koje proširuju mogućnosti Zabbix sustava za nadzor. Struktura projekta prikazana je na slici 5.1.



Slika 5.1. Blok dijagram projektnog zadatka.

5.1. Implementacija Zabbix sustava

Nakon stvaranja 4. virtualna računala potrebno ih je povezati u klaster, gdje su dva računala masteri, dva slaveovi. Na master i slave čvorove potrebno je instalirati i podesiti zabbix agenta. Zabbix ima javno dostupan repozitorij gdje možemo pronaći verzije za različite distribucije Linux operacijskog sustava. Zabbixov direktorij se nalazi na sljedećem linku https://repo.zabbix.com/zabbix. U projektu je korišten CentOS 7.4 operacijski sustav te je verzija Zabbixa koju koristimo 3.4, za što je potrebno dohvatiti odgovarajuću verziju. Url putanja do repozitorija na kojem se nalazi Zabbix poslužitelj verzije 3.4 koji odgovara našem operacijskom sustavu prikazana je slikom 5.2.
https://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-2.el7.noarch.rpm



Slika 5.2. Url putanja do Zabbix paketa s verzijom 3.4.2 za RHEL 7 operacijski sustav.

5.1.1. Instalacija poslužitelja

Za instalaciju poslužitelja potrebno je prvo dohvatiti i instalirati zabbix repozitorij koji sadrži konfiguracijske datoteke koje su potrebne yum paketnom menadžeru za instalaciju. Dohvaćanje datoteke sa konfiguracijom postižemo sljedećom naredbom:

```
rpm -ivh http://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-2.el7.noarch.rpm
```

Sljedeći korak je instalacija Zabbix poslužitelja s MySQL bazom podataka:

```
yum install zabbix-server-mysql zabbix-web-mysql
```

Nakon uspješne instalacije potrebno je pokrenuti bazu podataka:

```
systemctl start mariadb
```

Kako bismo instalirali bazu podataka na poslužitelju potrebno je pokrenuti sigurnu MySQL instalaciju koja omogućuje postavljanje lozinke za root korisnika, onemogućuje pristup root korisniku van lokalne mreže, uklanjanja anonimne korisnike, te uklanjanje testne baze podataka kojoj može pristupiti bilo koji korisnik. Za sigurnu instalaciju potrebno je pokrenuti skriptu:

```
mysql_secure_installation
```

Nakon sigurne instalacije potrebno je postaviti password za MySQL root korisnika:

```
Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Zatim je potrebno prijaviti se na MySQL bazu kao root korisnik:

```
mysql -uroot -plozinka
```

Instalirane baze podataka vidimo izvšenjem naredbe:

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)
```

Nakon što smo se prijavili na MySQL poslužitelj ne koristimo više bash nego smo se prebacili u *MariaDB [(none)]>* shell gdje je potrebno staviti točku zarez na kraj naredbe kako bismo izvršili MySQL upit.

Stvaranje daze podataka postižemo izvršenjem naredbe:

```
create database zabbix character set utf8 collate utf8_bin;
```

Zatim ćemo stvoriti novog lokalnog korisnika “zabbix” sa svim privilegijama unutar novo stvorene baze s odgovarajućom lozinkom:

```
grant all privileges on zabbix.* to zabbix@localhost identified by 'lozinka';
```

Te ćemo na posljetku izaći iz mariadb shella:

```
MariaDB [(none)]> exit;
Bye
```

Nakon stvaranja baze potrebno je učitati inicijalnu schemu koja će definirati izglede tablica, relacije među njima, tipove podata za pojedino polje u tablici te definiranje primarnih i stranih ključeva:

```
zcat /usr/share/doc/zabbix-server-mysql-3.4.2/create.sql.gz | mysql -uzabbix -p zabbix
```

Nakon čega će nas zatražiti lozinku koju smo postavili za zabbix korisnika.

5.1.2. Konfiguracija baze podataka za Zabbix poslužitelja

Da bi se Zabbix poslužitelj uspješno povezao sa MariaDB bazom podataka potrebno je izmjeniti odgovarajuće postavke unutar konfiguracijske datoteke na Zabbix poslužitelju koja se nalazi na putanji /etc/zabbix/zabbix_server.conf.

```
vi /etc/zabbix/zabbix_server.conf
```

Potrebno je izmjeniti IP adresu ili ime domaćina (engl. *Hostname*), naziv baze, naziv korisnika, te lozinku za navedenog korisnika.

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=lozinka
```

SELinux (Security-Enhanced Linux) je sigurnosna arhitektura integrirana unutar Linux kernela koja definira prava pristupa svakog korisnika, aplikacije, procesa i datoteka u sustavu. Zbog SELinuxa i dodjele prava pristupa potrebno je izvršiti sljedeće naredbe:

```
getenforce
setenforce 0
setsebool -P httpd_can_connect_zabbix on
```

Naredbom getenforce sustav vraća trenutno postavljeni način rada SELinux modula. SELinux modul može poprimiti jedno od 3 stanja, a to su: Permissive, Enforcing i Disabled. Postavljanje SELinux modula u "popustljiv" (engl. *Permissive*) način rada postižemo naredbom setenforce kojoj predajemo 0 kao argument.

Nakon završetka konfiguracije potrebno je pokrenuti Zabbix poslužitelja i postaviti da se automatski pokreće s operacijskim sustavom:

```
systemctl start zabbix-server
systemctl enable zabbix-server
```

5.1.3. PHP konfiguracija za Zabbix mrežnog sučelja

Apache konfiguracijska datoteka za Zabbix frontend se nalazi na */etc/httpd/conf.d/zabbix.conf* putanji. Većina postavki je automatski konfigurirana, međutim potrebno je otkomentirati "date.timezone" te postaviti odgovarajuću vremensku zonu. Lista odgovarajućih vremenskih zona možemo pronaći na: http://php.net/manual/en/timezones.php
Nakon što pronađemo odgovarajuću vremensku zonu potrebno je otvoriti konfiguracijsku datoteku te ju izmijeniti s odgovarajućim vrijednostima:

```
vi /etc/httpd/conf.d/zabbix.conf
```

```
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
```

```
php_value max_input_time 300
php_value always_populate_raw_post_data -1
php_value date.timezone Europe/Zagreb
```

Po završetku konfiguracije potrebno je pokrenuti Apache web server:

```
systemctl start httpd
```

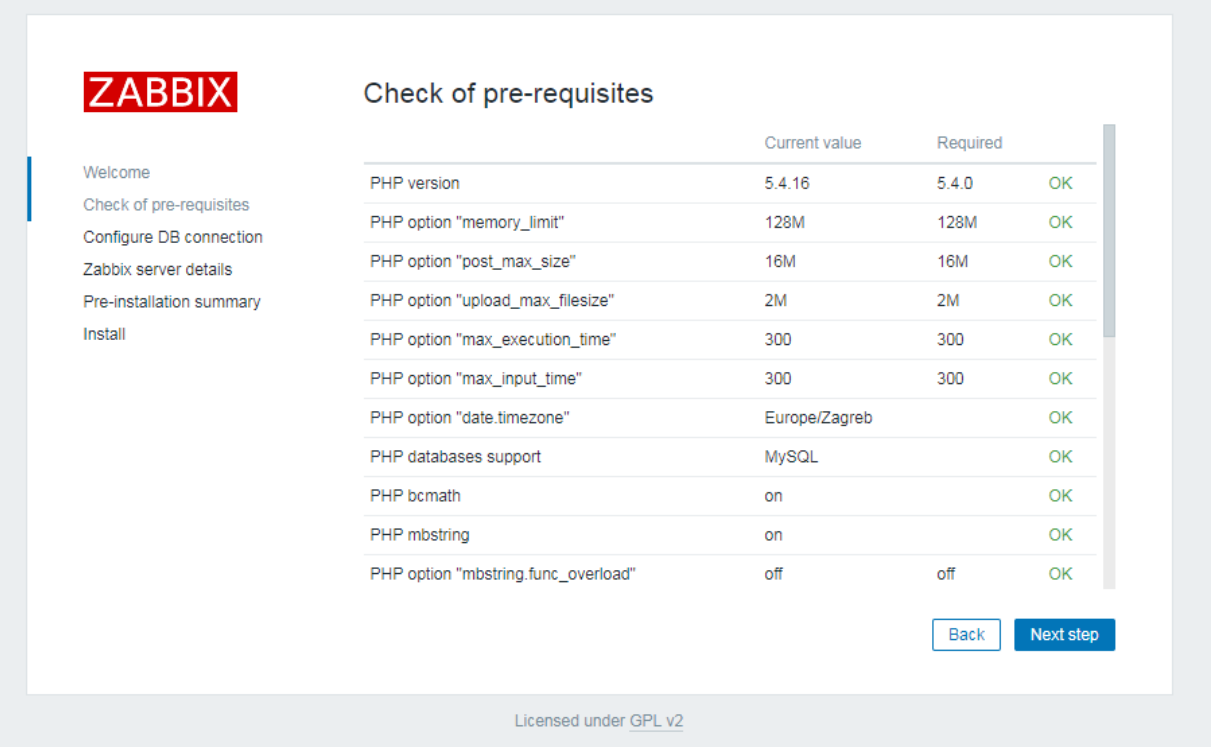
5.1.4. Instalacija mrežnog sučelja

Unutar web preglednika otvorimo Zabbix UI : http://<server_ip>/zabbix

Nakon odlaska na URL-a pojavit će nam se instalacija mrežnog sučelja.

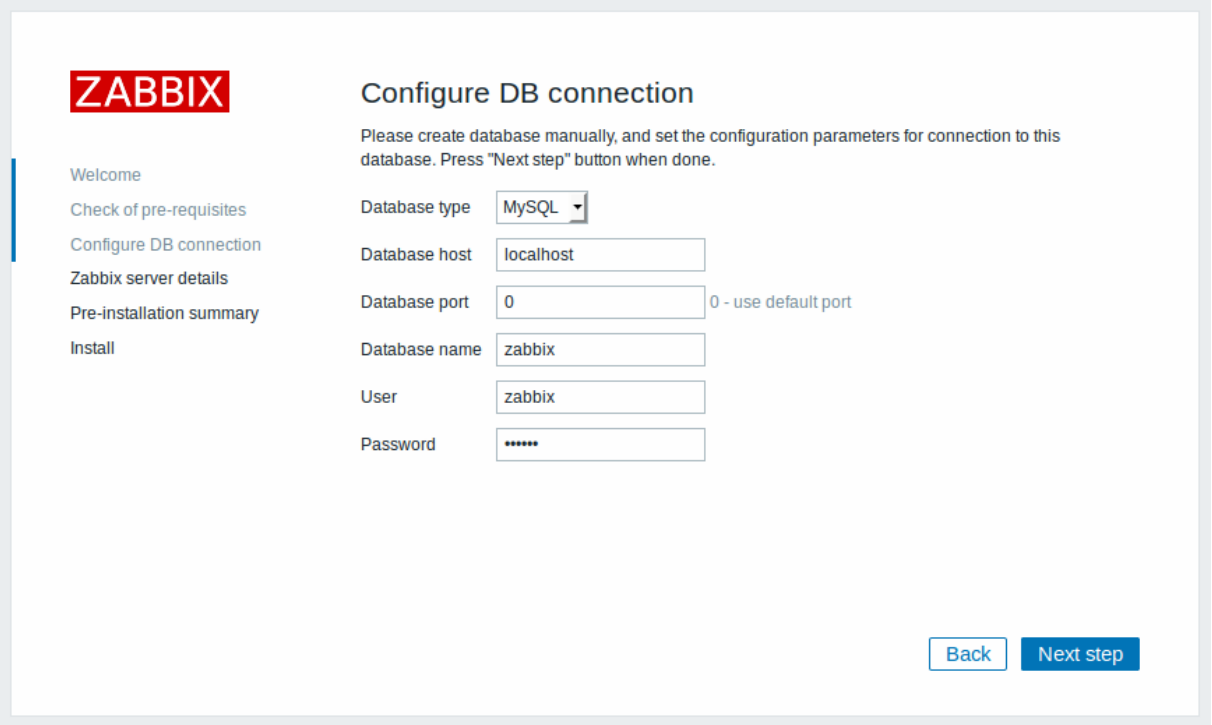
Slika 5.3. Slika ekrana prikazuje Zabbix mrežno korisničko sučelje.

Sljedeći korak je provjera preduvjeta za Zabbix. Potrebno je zadovoljiti sve uvijete kako bismo mogli nastaviti dalje s konfiguracijom.



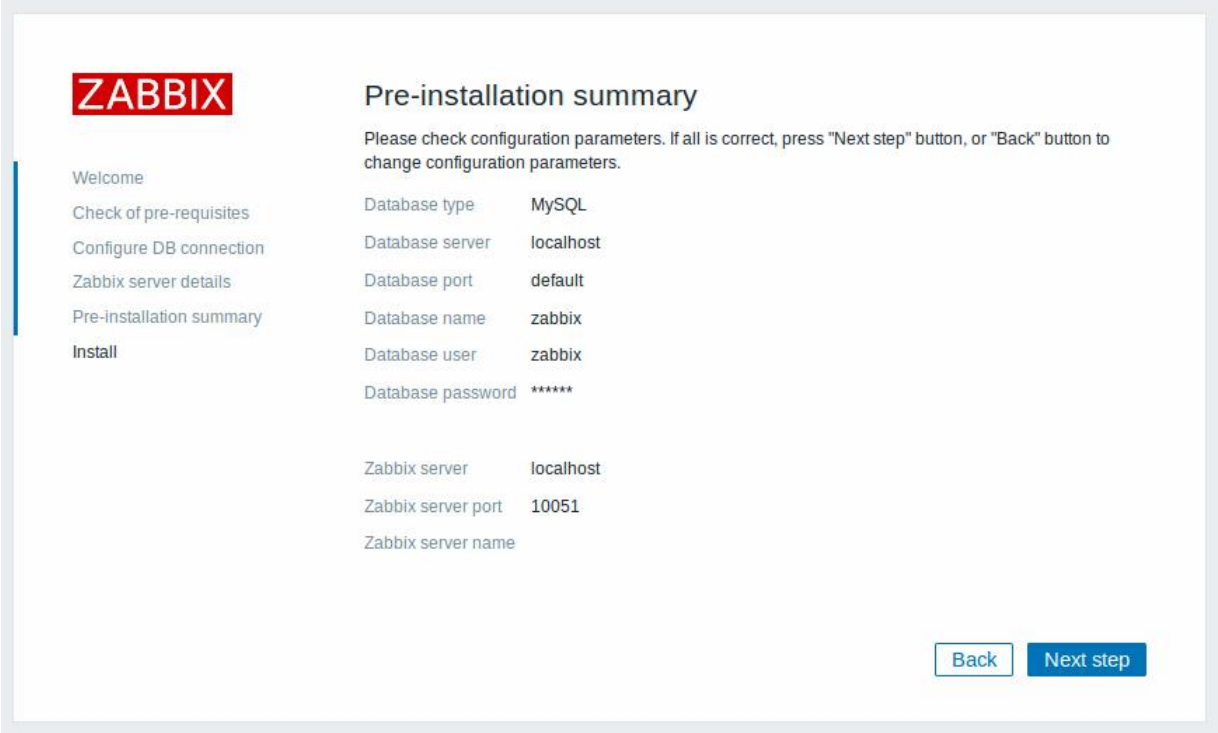
Slika 5.4. Slika ekrana prikazuje provjeru preduvijeta.

Zatim je potrebno postaviti pristupne podatke za povezivanje sa MariaDB bazom podataka:



Slika 5.5. Slika ekrana prikazuje povezivanje baze podataka s mrežnim korisničkim sučeljem.

Te na posljetku pregled postavki prije instalacije:



Slika 5.6. Slika ekrana prikazuje pregled postavki prije instalacije.

Nakon uspješne konfiguracije i instalacije potrebno je prijaviti se na mrežno korisničko sučelje. Zadano korisničko ime je “Admin”, a lozinka “zabbix”.

5.1.5. Instalacija proxy-a

Za instalaciju proxy-ja potrebno je prvo dohvatiti i instalirati zabbix repozitorij koji sadrži konfiguracijske datoteke koje su potrebne yum paketnom menadžeru za instalaciju. Dohvaćanje datoteke sa konfiguracijom postižemo sljedećom naredbom:

```
rpm -ivh http://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-2.el7.noarch.rpm
```

Nakon što smo dohvatili zabbix repozitorij potrebno je instalirati zabbix-proxy:

```
yum install zabbix-proxy
```

Te mariadb bazu podataka:

```
yum install mariadb
```

Nakon instalacije mariadb baze podataka potrebno je dozvoliti mariadb servisu pokretanje sa sustavom:

```
systemctl enable --now mariadb
```

Zatim se prijavimo u mariadb:

```
mysql -u root -p
```

Te nakon što se prijavimo potrebno je stvoriti novu bazu podataka i korisnika:

```
CREATE DATABASE zabbix CHARACTER SET utf8;
GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost' identified by 'lozinka';
FLUSH PRIVILEGES;
EXIT;
```

Gdje je naziv baze podataka “zabbix”, naziv korisnika “zabbix” te je lozinka baze podataka “lozinka”.

Potrebno je učitati default zabbix schemu u bazu podataka:

```
zcat /usr/share/doc/zabbix-proxy-mysql-3.4.2/schema.sql.gz | mysql -u zabbix -plozinka
zabbix
```

Kao i kod Zabbix poslužitelja potrebno je izmijeniti konfiguracijsku datoteku za proxy. Konfiguracijska datoteka se nalazi na /etc/zabbix/zabbix_proxy.conf putanji, te ju je potrebno otvoriti s nekim od uređivača teksta. Parametri koje trebamo izmjeniti su:

```
Server=<ip-of-zabbix-server>
DBName=zabbix
DBUser=zabbix
DBPassword=lozinka
```

Nakon završetka konfiguracije baze podataka i proxy-a potrebno je omogućiti proxy-u pokretanje sa sustavom te ga pokrenuti:

```
systemctl enable --now zabbix-proxy
systemctl start zabbix-proxy
```

Na posljetku je potrebno otvoriti portove na vatrozidu (engl. *Firewall*):

```
firewall-cmd --permanent --add-port=10050-10053/tcp
firewall-cmd --permanent --add-port=10050-10053/udp
firewall-cmd --permanent --add-port=162/tcp
firewall-cmd --permanent --add-port=162/udp
```

5.1.6. Instalacija Zabbix agenta

Svako računalo na kojem se vrši nadzor potrebno je instalirati Zabbix agenta. Da bi uspješno instalirali Zabbix agenta potrebno je dohvatili i instalirati zabbix repozitorij:

```
rpm -ivh http://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-
2.el7.noarch.rpm
```

Te instalirati zabbix agenta, sender i get:

```
yum install zabbix-agent zabbix-sender zabbix-get
```

Nakon instalacije potrebno je unutar konfiguracijske datoteke agenta promijeniti Server, ServerActive te Hostname:

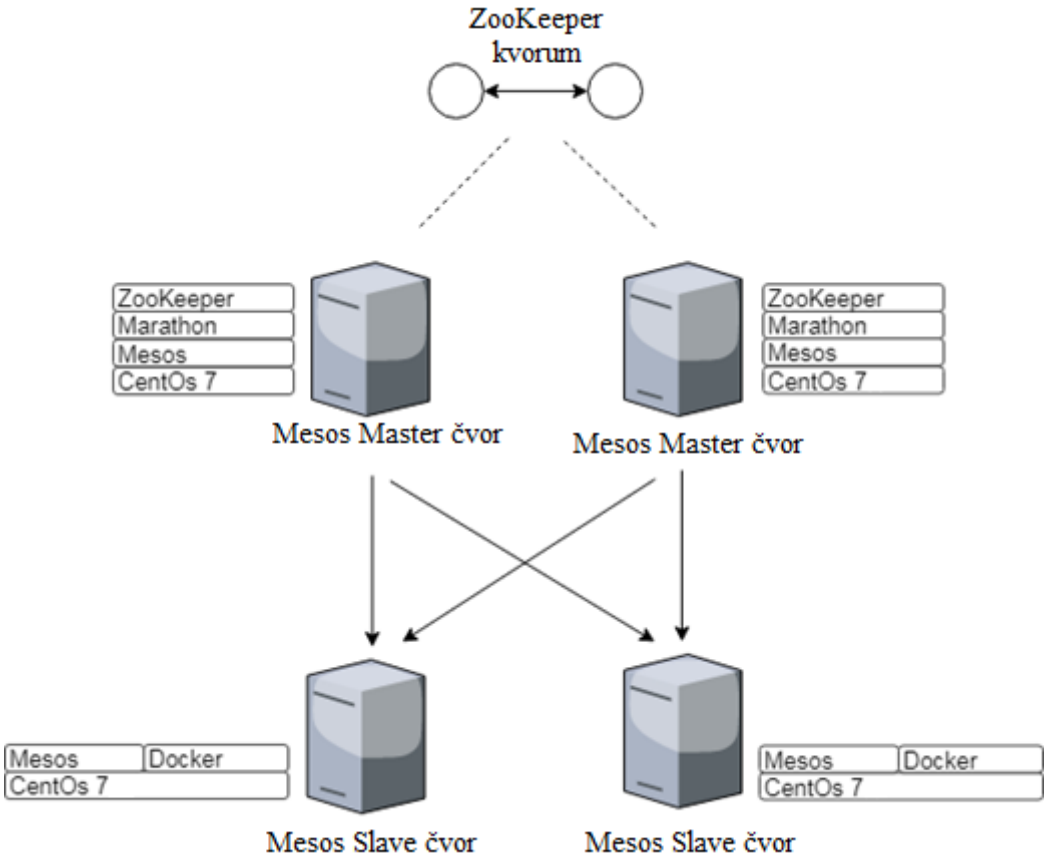
```
vi /etc/zabbix/zabbix_agentd.conf
Server=<proxy_ip>
ServerActive=< proxy_ip >
Hostname=<naziv_računala>
```

Nakon završetka konfiguracije potrebno je omogućiti agentu-u pokretanje sa sustavom te ga pokrenuti:

```
systemctl enable zabbix-agent
systemctl start zabbix-agent
```

5.2. Implementacija Mesosa

Mesos struktura je ostvarena tako da imamo 2 master i 2 slave čvora. Master čvorove nadzire ZooKeeper koji određuje tko je vođa (engl. *Leader*), jer samo vođa komunicira s Zabbix proxy-em. Izgled strukture prikazan je na slici 5.7.



Slika 5.7. Struktura Mesos klastera.

5.2.1. Instalacija Master čvorova

Za instalaciju mesos mastera potrebno je prvo dohvatiti mesos repozitorij koji sadrži konfiguracijske datoteke potrebne yum paketnom menadžeru za instalaciju, te instalirati mesos, marathon i zookeeper servise odnosno aplikacije:

```
yum install http://repos.mesosphere.io/el/7/noarch/RPMS/mesosphere-el-repo-7-1.noarch.rpm
```

```
yum install mesos marathon
yum install mesosphere-zookeeper
```

Za svaki master čvor trebamo postaviti „myid“ datoteku koja se nalazi na putanji /var/lib/zookeeper, unutar koje definiramo „id“ master čvora. Pomoću sljedeće naredbe definiramo prvi master čvor s identifikatorom 1.

```
echo 1 > /var/lib/zookeeper/myid
```

Svakom master čvoru potrebno je postaviti IP adresu unutar hostname datoteke koje se nalazi na putanji /etc/mesos-master/hostname:

```
echo '<ip_adresa_master_čvora>' > /etc/mesos-master/hostname
```

5.2.2. Instalacija Slave čvora

Za instalaciju slave čvorova potrebno je dohvatiti mesos repozitorij i instalirati mesos:

```
yum install http://repos.mesosphere.io/el/7/noarch/RPMS/mesosphere-el-repo-7-
1.noarch.rpm
yum -y install mesos
```

Nakon instalacije potrebno je konfigurirati mesos s dockerom te podesiti vrijeme registracije (engl. *Registration timeout*):

```
echo 'docker,mesos' > /etc/mesos-slave/containerizers
```

```
echo '5mins' > /etc/mesos-slave/executor_registration_timeout
```

Zatim je potrebno dodati zookeeper konfiguraciju do mesos-master čvorova na svakom slave čvoru:

```
echo 'zk:// <master_1_Ip>:2181,<master_2_ip>:2181/mesos' > /etc/mesos/zk
```

Na posljetku na svakom slave čvoru isključimo mesos-master servis te ponovno pokrenemo mesos-slave servis:

```
systemctl disable mesos-master
systemctl restart mesos-slave
```

5.2.3. Instalacija Docker-a na Slave čvorovima

Kako bismo omogućili postavljanje kontejnera na slave čvorove potrebno je dohvatiti instalacijski paket za docker te ga instalirati:

```
wget https://yum.dockerproject.org/repo/main/centos/7/Packages/docker-engine-1.13.0-1.el7.centos.x86_64.rpm

wget https://yum.dockerproject.org/repo/main/centos/7/Packages/docker-engine-selinux-1.13.0-1.el7.centos.noarch.rpm

yum install docker-engine-1.13.0-1.el7.centos.x86_64.rpm docker-engine-selinux-1.13.0-1.el7.centos.noarch.rpm
```

Nakon instalacije potrebno je konfigurati docker na sljedeći način:

```
vi /etc/systemd/system/docker.service.d/docker.conf
```

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -g=/opt/app/docker -H tcp://0.0.0.0:2375 -H
unix:///var/run/docker.sock
```

Te na posljetku pokrenuti Docker servis:

```
systemctl start docker
```

Mesosu i Marathonu pristupamo tako da ih otvorimo unutar mrežnog preglednika:

Mesos: http://<ip_master_noda>:5050

Marathon: http://<ip_master_noda>:8080

5.2.4. Konfiguracija ZooKeeper servisa

Za konfiguraciju zookeeper servisa potrebno je dodati svaki master čvor unutar konfiguracijske datoteke. Dodajemo ih tako da dodamo izraz server.myid= <ip adresa master noda >:2888:3888 unutar konfiguracijske datoteke.

```
vi /etc/zookeeper/conf/zoo.cfg
```

server.1=<IP_prvog_master_noda>:2888:3888

server.2=<IP_drugog_master_noda>:2888:3888

Nakon uspješne konfiguracije potrebno je pokrenuti zookeeper servis:

```
systemctl start zookeeper
```

Nakon što smo pokrenuli zookeeper potrebno ga je povezati s Mesos master čvorovima ako želimo da ih zookeeper nadzire. Dodat ćemo listu master čvorova (IP adrese s portovima) razdvojene zarezom unutar /etc/mesos/zk:

```
echo 'zk://<master_1_Ip>:2181,<master_2_ip>:2181/mesos' > /etc/mesos/zk
```

Potrebno je definirati kvorum (engl. *Quorum*) vrijednost za Mesos master čvorove. S obzirom na to da imamo dva master čvora potreban nam je barem jedan kako bi mogao donijeti odluku o trenutnoj akciji.

```
echo 1 > /etc/mesos-master/quorum
```

Kako bismo povezali zookeeper s marathonom potrebno je izmijeniti /mesos u /marathon unutar /etc/marathon/conf/zk:

```
vi /etc/marathon/conf/zk
```

```
zk://<master_1_Ip>:2181,<master_2_ip>:2181/marathon
```

Te na svakom master čvoru unutar datoteke marathon na putanji /etc/default/marathon dodati:

```
MARATHON_MASTER=zk://<master_ip>:2181/mesos
MARATHON_ZK=zk://<master_ip>:2181/marathon
```

Na posljetku je potrebno zaustaviti mesos-slave servis, te ponovno pokrenuti mesos-master servis, te ponovno pokrenuti Marathon kako bi se aktivirale promjene konfiguracije:

```
systemctl stop mesos-slave
systemctl restart mesos-master
systemctl restart marathon
```

5.2.5. Skripte

Kako bismo mogli podatke iz Marathon API-ja proslijediti Zabbix proxy-u putem Zabbix agenta potrebno je napisati skriptu. Napisati ćemo skripte marathon_zabbix_helper te mesos_leader_check. Skripte je potrebno postaviti na putanju /etc/zabbix/scripts na svakom

master čvoru, te unutar /etc/zabbix/zabbix_agentd.conf postaviti UserParameter tako da poziva skriptu:

```
UserParameter=marathon_zabbix_helper[*],/etc/zabbix/scripts/marathon_zabbix_helper.sh $1
UserParameter=mesos_leader_check[*], /etc/zabbix/scripts/mesos_leader_check.sh $1
```

Programski kod Marathon_zabbix_helper.sh skripte

```
#!/bin/bash

output=$(curl -sk "http://<IP_master_1>:8080/v2/apps/$1" | egrep -o
'"tasksHealthy":[0-9]{1,3}')

if [ -z "$output" ]; then
   output=$(curl -sk "http://<IP_master_2>:8080/v2/apps/$1" | egrep -o
'"tasksHealthy":[0-9]{1,3}')
   echo $output | awk -F':' '{print $2}'
else
   echo $output | awk -F':' '{print $2}'
fi
```

Programski kod Mesos_leader_check.sh skripte

```
#!/bin/bash

curl -sk http://$1:5050/master/state | python -m json.tool 2> /dev/null | grep leader
| grep -v leader_info | wc -l
```

5.3. Implementacija Dockeriziranih aplikacija pomoću Marathona

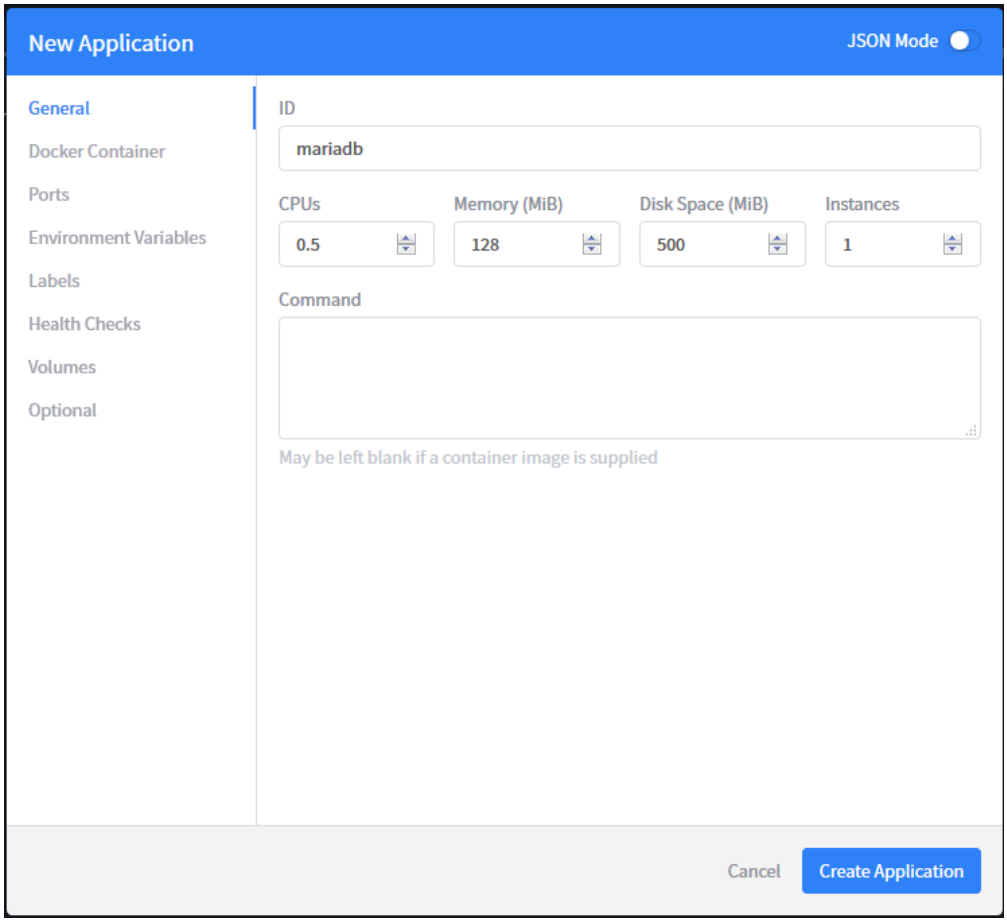
Kako bismo implementirali Dockeriziranu aplikaciju potrebno ju je prvo stvoriti ili koristiti već stvorenu aplikaciju. Na https://hub.docker.com je moguće pronaći dockerizirane aplikacije. Stavranje aplikacije s docker hub-a izvodi se pomoću naredbu za povlačenje (pull):

```
docker pull mariadb
```

Kako bismo pristupili korisničkom sučelju Marathona potrebno je unutar mrežnog preglednika pristupiti master čvoru tako da upišemo IP adresu master čvora koji je ujedno i vođa (engl. *Leader*) te port 8080:

http://<master_ip >:8080

Nakon odabira „Create application“ pojavit će se prozor:



Slika 5.8. Slika ekrana prikazuje korisničko sučelje za stvaranje aplikacije (engl. *Create application*).

ID: naziv kontejnera

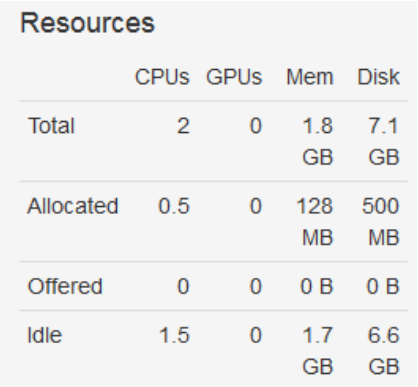
CPUs: 0.01-N (CPU koji želimo dodijeliti kontejneru)

Memory: 32-N (Memorija koji želimo dodijeliti kontejneru)

Disk Space: 0-N (Prostor na disku koji želimo dodijeliti kontejneru)

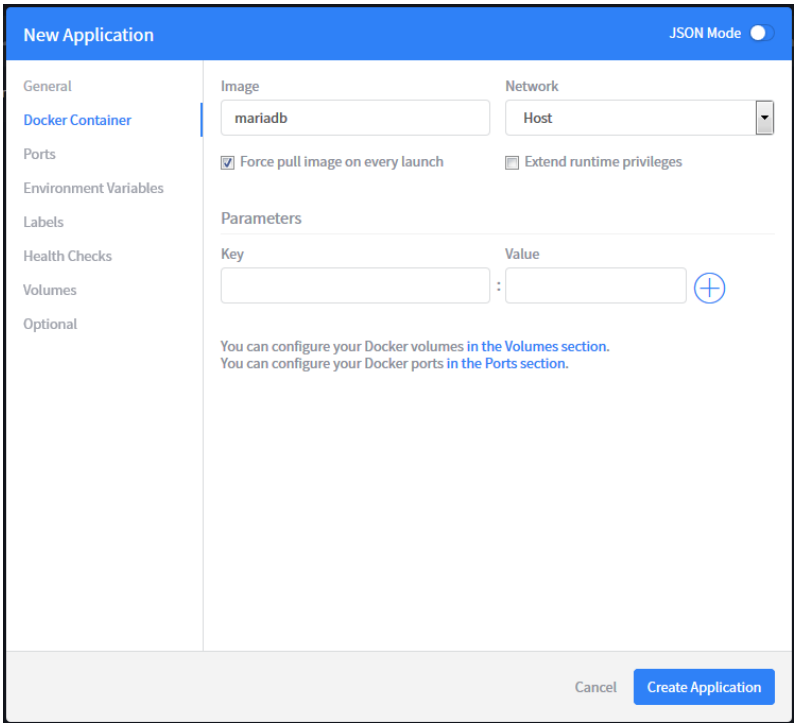
Instances: 0- N (Broj kontejnera koje želimo stvoriti)

Vrijednosti N možemo provjeriti unutar Mesos-a:



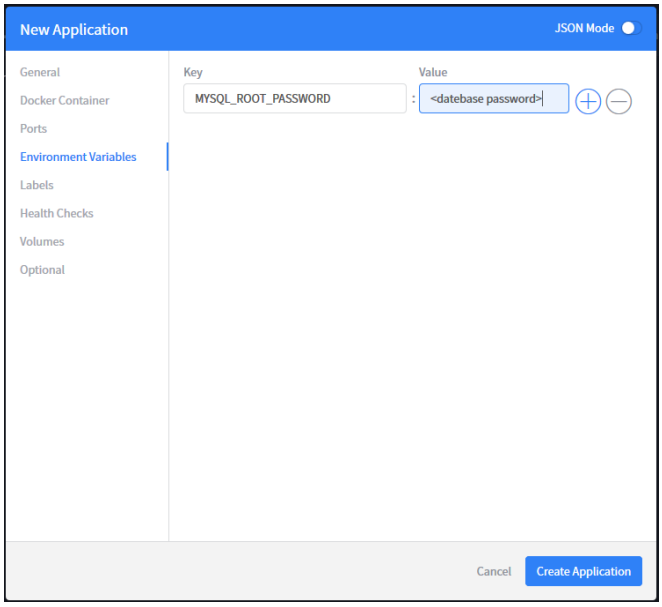
Slika 5.9. Slika ekrana prikazuje dostupne resurse.

Unutar kartice (engl. *Tab*) "Docker Container" unosimo naziv slike (engl. *Image*) iz kojeg želimo stvoriti kontejner (docker pull **mariadb**), te označimo "Force pull image on every launch" kako bi kontejner ostao ažuriran.



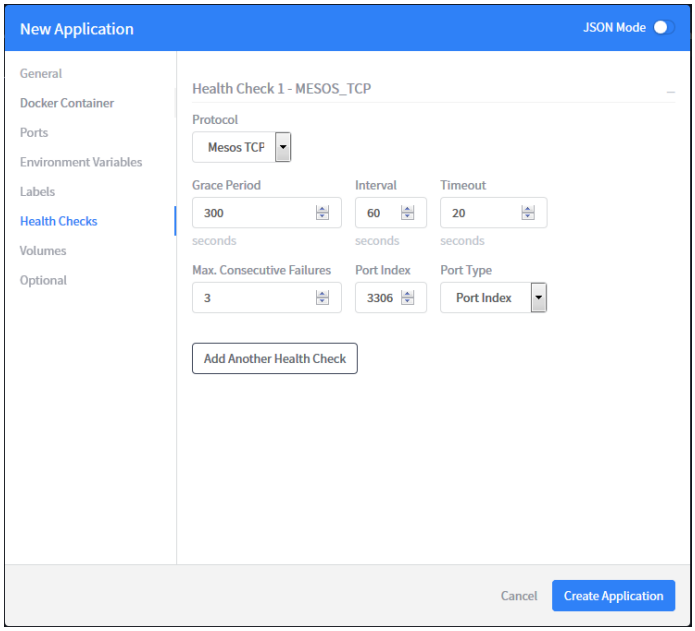
Slika 5.10. Slika ekrana korisničkog sučelja prikazuje „Docker Container" karticu.

Unutar „Enviornment Variables” kartice postavljamo vrijednost varijabli okoline (neke slike imaju obvezne varijable okoline koje su potrebne za normalan rad kontejnera. Varijable okoline možemo pronaći na: https://hub.docker.com/_/mariadb/)



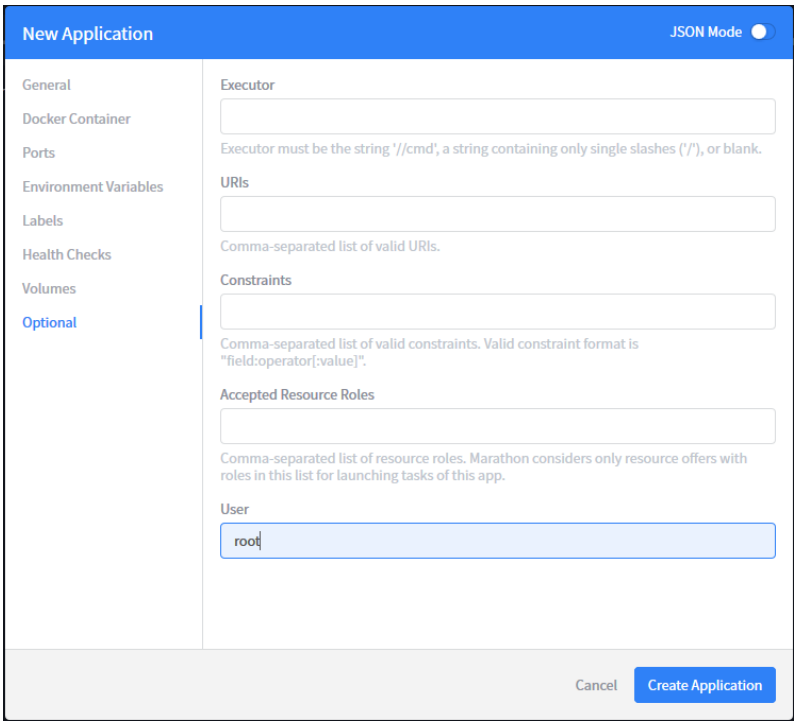
Slika 5.11. Slika ekrana korisničkog sučelja prikazuje karticu varijable okoline.

Provjera zdravlja servisa odnosno aplikacija (engl. *Health Check*) nije obvezna, ali je jako korisna za brzi uvid u stanje kontejnera. Na slici 5.12. vidimo konfiguraciju provjere zdravlja za mariadb kontejner. Ako je sve dobro dobit ćemo odziv na portu 3306 te će status kontejnera biti zdrav.



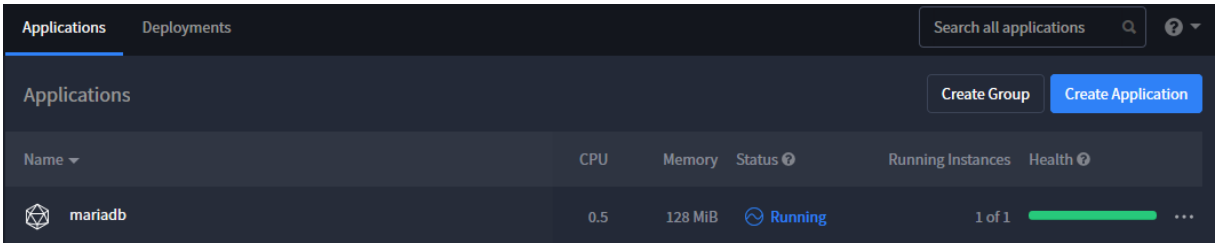
Slika 5.12. Slika ekrana korisničkog sučelja prikazuje karticu provjere zdravlja za mariadb kontejner.

Unutar „Optional“ kartice unosimo korisnika pomoću kojega pokrećemo kontejner:



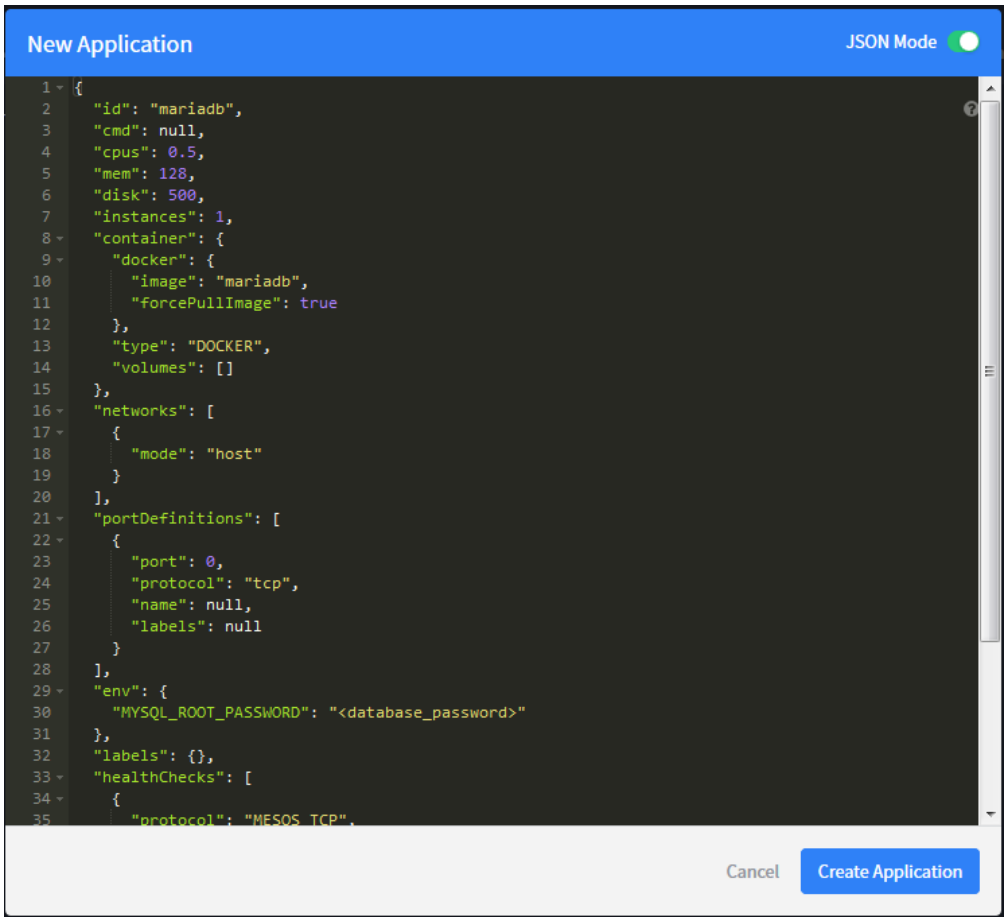
Slika 5.13. Slika ekrana **k**orisničkog sučelja prikazuje karticu dodatne opcije.

Nakon konfiguracije odaberemo „Create Application“ dugme, te će nakon uspješnog konfiguriranja biti vidljiv pokrenuti kontejner:



Slika 5.14. Slika ekrana prikazuje kontejner nakon konfiguracije.

Napredno konfiguriranje kontejnera se radi pomoću JSON Mode u kojem ručno unesemo konfiguraciju za odgovarajuće parametre u JSON strukturi:



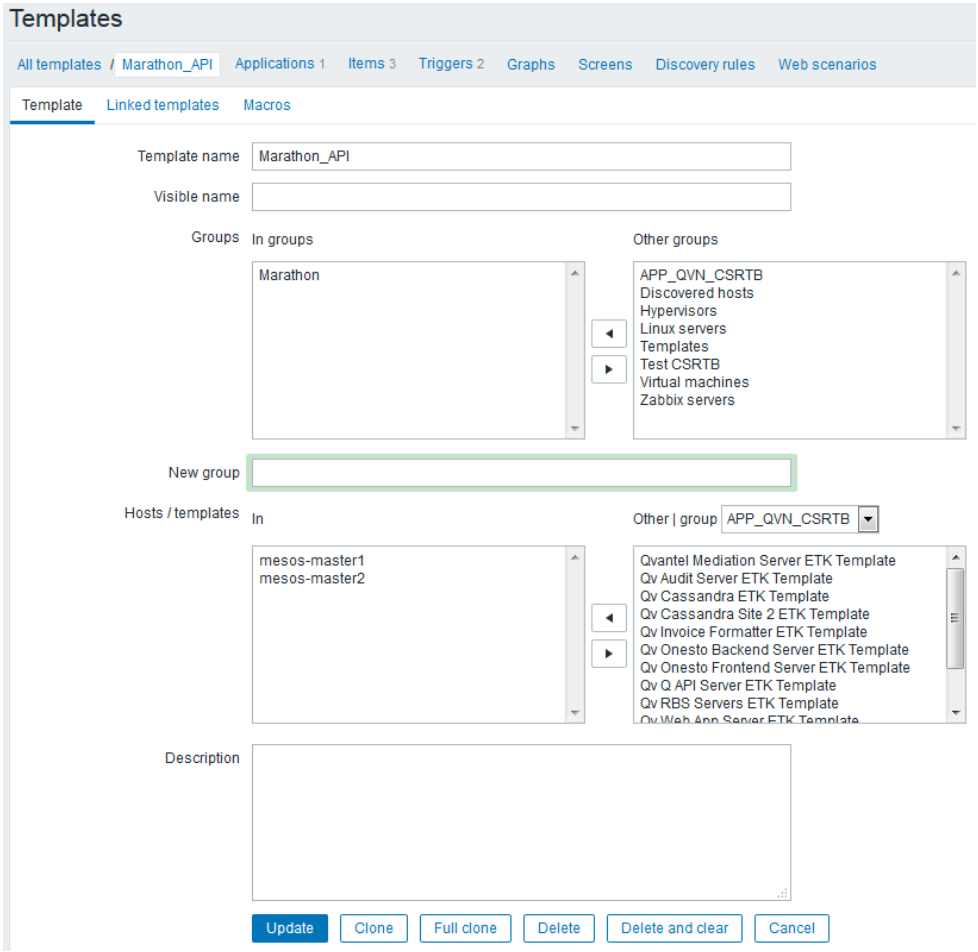
Slika 5.15. Slika ekrana prikazuje JSON Mode.

5.4. Postavljanje nadzora unutar Zabbix mrežnog korisničkog sučelja

Kako bi mogli nadzirati sustav potrebno je stvorit predložak (ili iskoristi postojeći predložak koji nadzire stvari koje nas zanimaju), stavke koje želimo nadzirati te na njih postaviti okidače koji će oglasiti alarm u slučaju da neka od stavki poprimi ne željeno stanje.

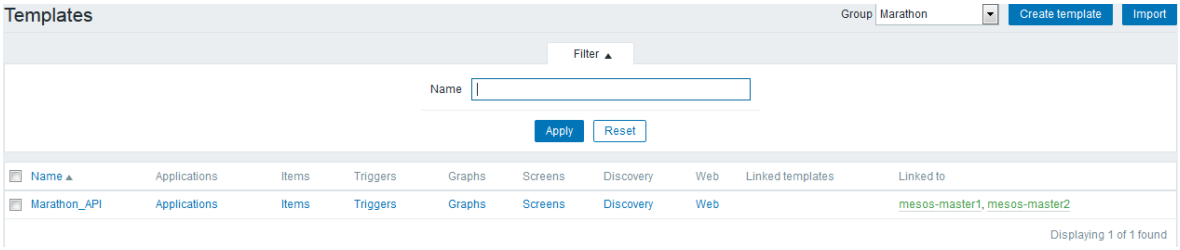
5.4.1. Stvaranje novog predloška

Za stvaranje novog predloška potrebno je upisati naziv predloška, grupu, te dodati hostove na koje želimo primijeniti predložak.



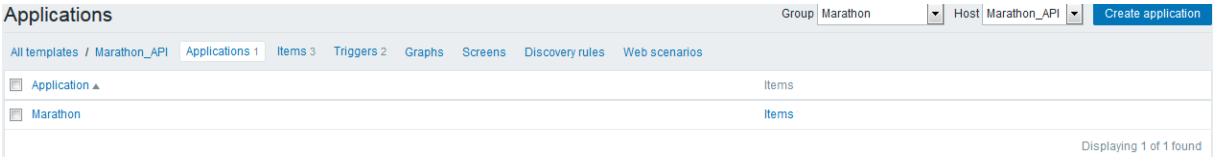
Slika 5.16. Slika ekrana prikazuje stvaranje novog predloška pomoću zabbix mrežnog korisničkog sučelja.

Nakon uspješnog stvaranja vidjet ćemo novo stvoreni predložak



Slika 5.17. Slika ekrana prikazuje stvorene predloške.

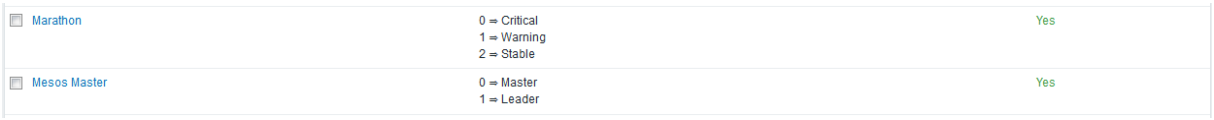
Zbog lakšeg nadzora stvorit ćemo aplikaciju za koju je potrebno samo upisati željeni naziv.



Slika 5.18. Slika ekrana prikazuje stvorene aplikacije.

5.4.2. Stvaranje mape vrijednosti

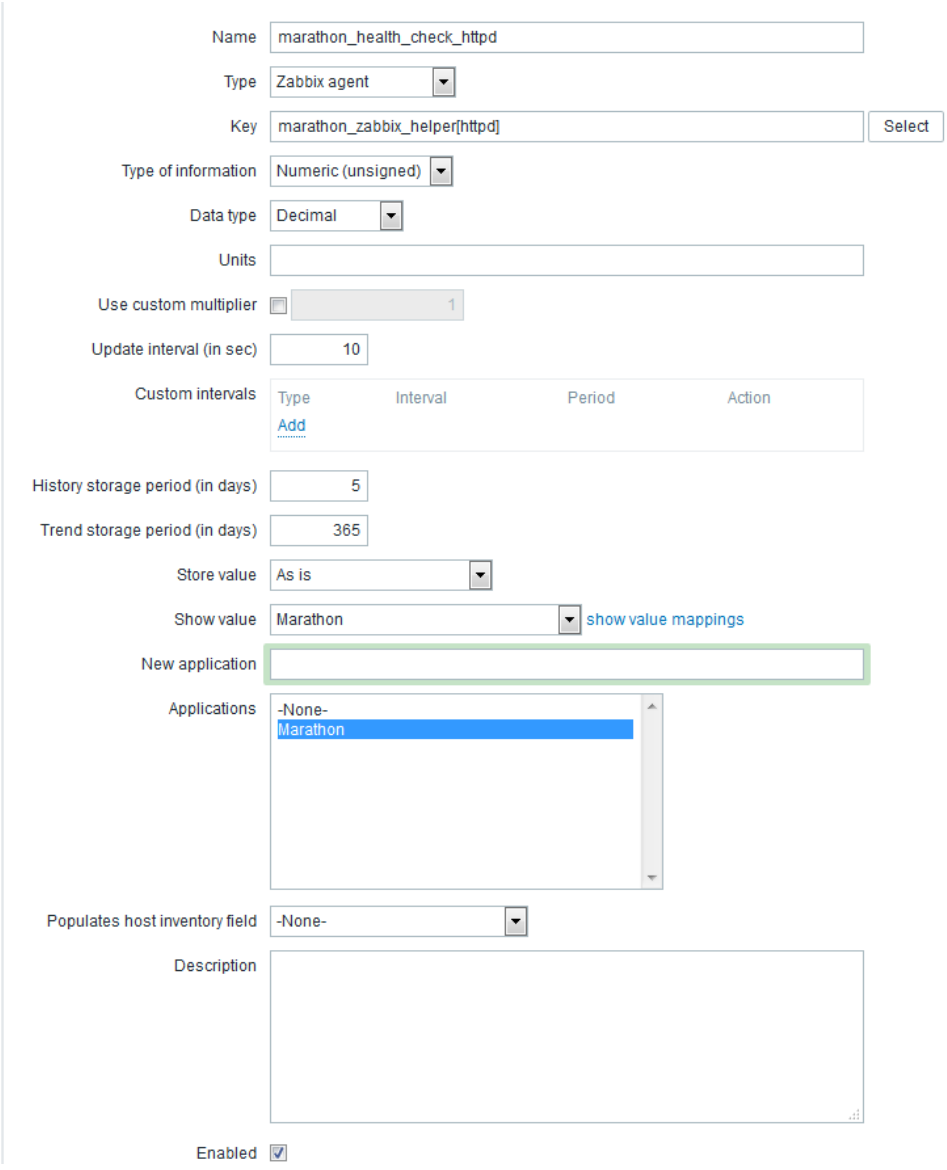
Stvaranje mape vrijednosti olakšava pregled informacija u zabbix korisničkom sučelju. Mape vrijednosti omogućuju pridruživanje sustavskih statusa u vrijednosti koje su operaterima razumljivije i čitljivije. Primjerice, brojčane vrijednosti aktivnih servisa povezujemo sa razinama ozbiljnosti pogreške. Slika 5.19 prikazuje mapu vrijednosti za stanje aplikacije koje nadziremo pomoću Marathona i mapu vrijednosti za provjeru izbora vođe u Mesos klasteru.



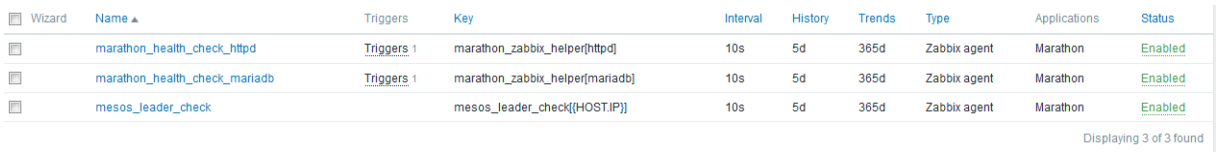
Slika 5.19. Slika ekrana prikazuje mape vrijednosti.

5.4.3. Stvaranje stavka za svaki mikro servis

Ključ (engl. *Key*) mora odgovarati korisničkim parametrima (engl. *UserParameter*) koji su definirani u datoteci zabbix_agentd.conf na /etc/zabbix/zabbix_agentd.conf putanji.



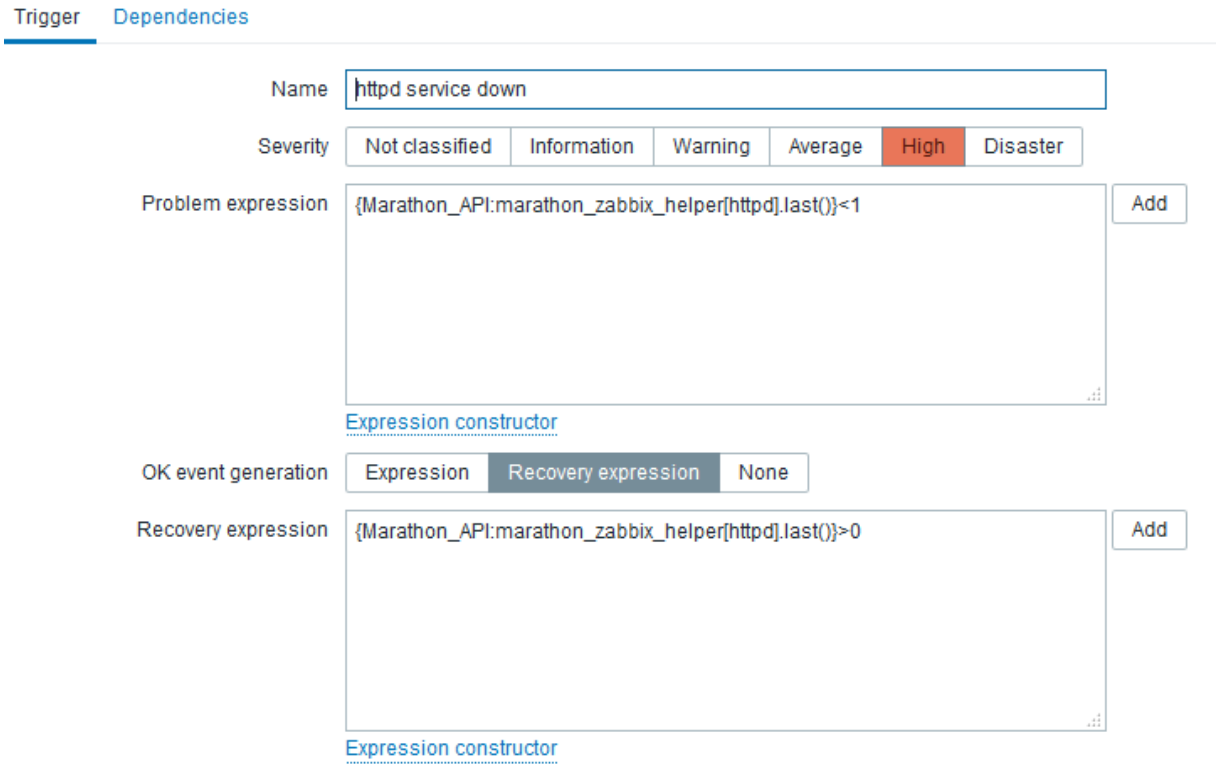
Slika 5.20. Slika ekrana prikazuje zaslon za stvaranje stavki.



Slika 5.21. Slika ekrana prikazuje stvorene stavke.

5.4.4. Stvaranje okidača

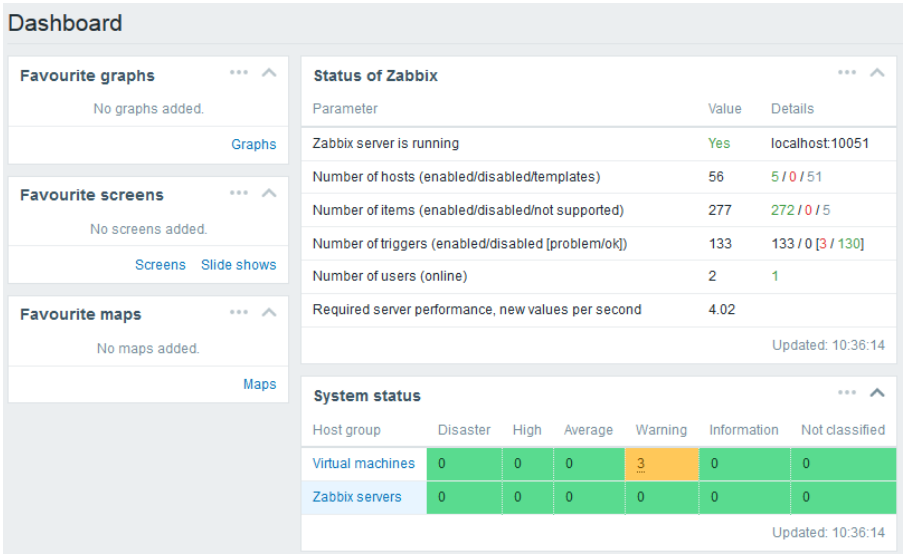
Za svaku stavku stvorit ćemo okidače koji će oglasiti alarm u slučaju da stavka poprimi ne željenu vrijednost.



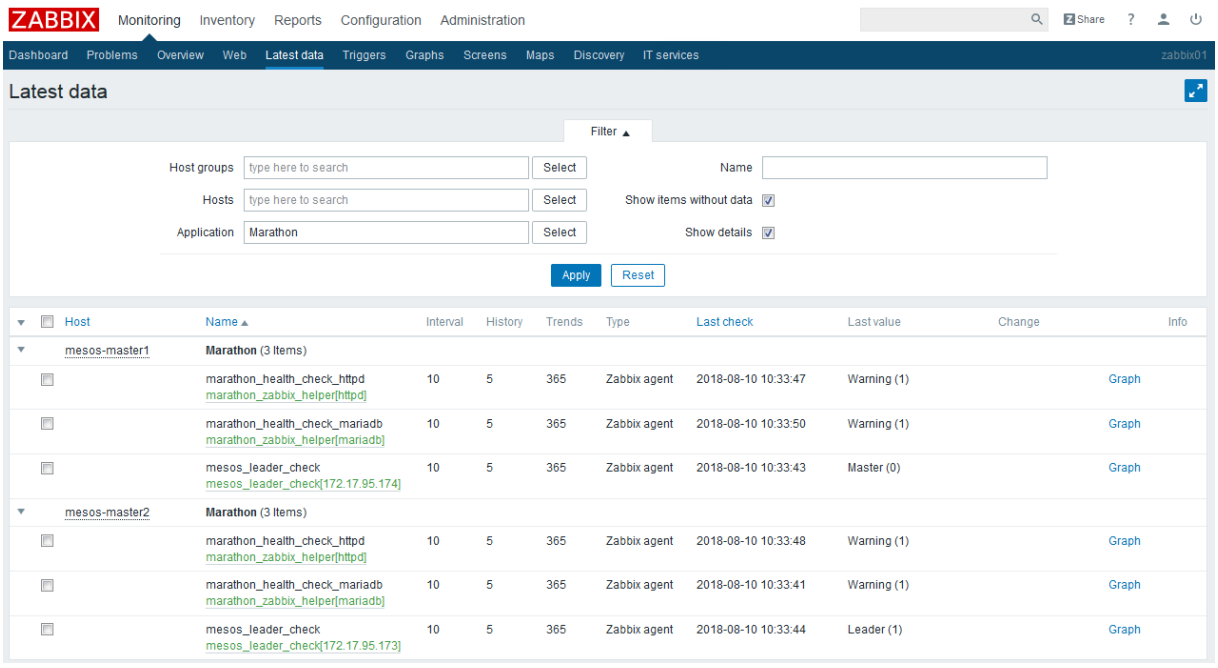
Slika 5.22. Slika ekrana prikazuje zaslona za stvaranje okidača.

5.4.5. Nadzor sustava

Nakon što je sve uspješno podešeno imamo nadzor nad sustavom koji možemo pratiti putem Zabbix mrežnog korisničkog sučelja. Na početnom zaslonu se nalaze općenite informacije o stanju sustavu kao što su: raspoloživost, broj hostova, broj stavki, broj okidača, broj korisnika te ostale sustavska stanja.



Slika 5.23. Slika ekrana prikazuje početni zaslon Zabbixa.



Slika 5.24. Slika ekrana prikazuje vrijednosti za definiranu aplikaciju Marathon.

6. ZAKLJUČAK

U današnje vrijeme računalne mreže podržavaju veliki broj aplikacija i servisa kao što su WWW (World Wide Web), bankarski i telekomunikacijski sustavi. Stoga se pojavila potreba nadzora mreža i servisa kako bi se povećala sigurnost prijenosa informacija, efikasnost i stabilnost čitavog sustava. Osnovu procesa upravljanja računalnim mrežama čine nadzor i praćenje. Zabbix je softver otvorenog koda za nadzor računalnih mreža, operacijskih sustava i aplikacija. Omogućava prikupljanje podataka pomoću nekoliko metoda koje uključuju SNMP i IPMI protokole, među platformske Zabbix agente i ugrađene funkcije kao što su provjera mrežnih servisa (dostupnost udaljenog računala i njegovih portova). Osim ugrađenih provjera Zabbix podržava stvaranje vlastitih provjera, koristeći korisničke parametre Zabbix agenta, koje pozivaju skripte pomoću kojih se proširuje mogućnost nadzora sustava. U diplomskom radu opisan je način nadzora baze podataka i mrežnog poslužitelja koji se nalaze u Docker kontejnerima uz pomoć platforme za orkestraciju Marathon. Docker se nalazi na slave čvorovima računalnog klastera pri čemu se nadzor obavlja putem master čvorova na kojima se nalaze Zabbix agenti, koji pokreću korisničke skripte koje dohvaćaju podatke o stanju aplikacija. Informacije o stanju aplikacija se šalju Zabbix poslužitelju na daljnju obradu pomoću Zabbix proxy-ja. U budućnosti nas očekuje sve veća prisutnost alata otvorenog koda u velikim sustavima. Razlog tome su cijena i zajednica koja radi na razvoju alata. Zabbix je primjer alata otvorenog koda koji se koristi za nadzor velikih sustava kao što su telekom operateri, banke, bolnice, istraživački centri, škole i mnogi drugi.

LITERATURA

[1] Osnove mrežnog nadzora [online], 10. lipnja 2018. dostupno na: https://www.solarwinds.com/basics-of-network-monitoring

[2] Dokumentacija zabbixa [online], 13. lipnja 2018. dostupno na: https://www.zabbix.com/documentation/3.2/manual/concepts

[3] Zabbix mrežno korisničko sučelje [online], 15. lipnja 2018. dostupno na: https://www.zabbix.com/zabbix_web_frontend

[4] Računalni klasteri [online], 22. lipnja 2018. dostupno na: https://www.techopedia.com/definition/6581/computer-cluster

[5] Način rada Apache Mesosa [online], 28. lipnja 2018. dostupno na: https://searchitoperations.techtarget.com/definition/Apache-Mesos

[6] Način rada Zookeepera [online], 7. srpnja 2018. dostupno na: https://www.ibm.com/analytics/hadoop/zookeeper

[7] Virutalizacija računalnih sustava [online], 12. srpnja 2018. dostupno na: https://www.techopedia.com/definition/660/operating-system-virtualization-os-virtualization

[8] Docker arhitektura [online], 17. srpnja 2018. dostupno na: https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/container-docker-introduction/docker-defined

[9] Docker arhitektura [online], 25. srpnja 2018. dostupno na: https://opensource.com/resources/what-docker

[10] Način rada Marathon orkestracijske platforme [online], 7. kolovoza 2018. dostupno na: https://mesosphere.github.io/marathon/

[11] J. Turnbull. *The Art of Monitoring*, Turnbull Press, 2016.

SAŽETAK

Računalne mreže su prisutne u svakom dijelu moderne civilizacije, te se zbog sve većih mreža pojavljuje potreba za njihovim nadzorom, optimizacijom i upravljanjem. Nadzor računalnog klastera i mreže ostvaren je pomoću Zabbix sustava. U diplomskom radu stvoren je računalni klaster kojega sačinjavanju četiri virtualne mašine, od kojih su dvije u Master načinu rada i dvije u Slave načinu rada. Klasteriranje virtualnih mašina ostvareno je pomoću Apache Mesos distribuiranog kernela, a upravljanje kontejnerima pomoću Marathon radnog okvira za orkestriranje kontejnera.

Ključne riječi: nadzor računalnih mreža, mrežno upravljanje, računalni klaster, platforma za orkestriranje, Zabbix, Docker

ABSTRACT

MONITORING APPLICATIONS AND SERVICES IN COMPUTER CLUSTER

Computer networks are present in every part of modern civilization, and because their growth, there is a need for monitoring, optimization and management. Monitoring of the computer cluster and the network is achieved with the Zabbix. In this paper we created a computer cluster composed of four virtual machines, where two are in Master mode and two in Slave mode. Virtual machine clustering was accomplished using the Apache Mesos distributed Kernel, and container management using the Marathon orchestration Framework.

Keywords: network monitoring, network management, computer cluster, orchestration platform, Zabbix, Docker

ŽIVOTOPIS

Josip Dumančić rođen je u Požegi, 27. listopada 1992. godine. Po završetku osnovne škole „Antun Kanižlić" upisuje „Tehničku školu" u Požegi, smjer tehničar za računarstvo. Nakon završene srednje škole, upisuje preddiplomski studij, smjer Računarstvo na Elektrotehničkome fakultetu u Osijeku (danas Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek). Nakon završetka preddiplomskog studija, upisuje diplomski studij, smjer Računalno inženjerstvo. Stručnu praksu je odradio u tvrtkama Atos iz Osijeka i Ericsson Nikola Tesla iz Zagreba.

PRILOZI