ZABBIX SERVER ZA PRAĆENJE SUSTAVA

Gatti, Frano

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:664743

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**



Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College





VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

ZABBIX SERVER ZA PRAĆENJE SUSTAVA

Frano Gatti

Zagreb, veljača 2020.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu datum.

Frano Gatti

Predgovor

Zahvaljujem se svim profesorima sistemskog inženjerstva na kvalitetno odrađenom poslu. Puno ste mi pomogli naučiti zanat koji će mi koristiti u životu. Izuzetno sam zahvalan na psihološkim savjetima i dobrim praksama za ravnotežu između poslovnog i privatnog života.

Prilikom uvezivanja rada,Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Opisane su ključne ideje centraliziranog praćenja sustava. Navedene su komponente sustava koje je bitno motriti. Sve je implementirano s *Linux* servisima za neprekidnu dostupnost. Svaki od servisa objašnjen je i navedeni su koraci konfiguracije. Definirani su koraci za podizanje i konfiguraciju *Zabbix* poslužitelja na *CentoOS* operacijskom sustavu. Glavni je cilj bio implementirati visoko dostupan poslužitelj za praćenje sustava. U virtualnom okruženju instalirani su servisi praćeni *Zabbixom*. Za testiranje funkcionalnosti podignuta je skupina virtualnih poslužitelja kakvi bi se podigli u produkcijskom okruženju. Prilikom pisanja *bash* i *powershell* skripti za svaku komandu napisano je obrazloženje u komentaru koji počinje znakom #.

Sadržaj

1. Uvod 1

2. Pravila, komponente i preporuke 2

2.1. Opis problema 2

2.2. Komponente *Zabbix* servera 4

2.3. Preporuke za implementaciju 4

3. Instalacija visoko dostupnog *Zabbix-servera* 6

3.1. Predradnje na Centos serveru 7

3.2. Visoke dostupnosti na razini *Zabbix* i *web-poslužitelja* 8

3.2.1. Konfiguracija visoke dostupnosti na razini aplikacije i *web-servera* 9

3.3. Visoka dostupnost na razini baze podataka 14

3.3.1. Konfiguracija *Galera* servisa visoke dostupnosti na razini baze podataka . 16

3.3.2. Konfiguracija virtualne *keepalive IP adrese* i *HAproxy* servisa 21

4. Podešavanje funkcionalnosti 25

4.1. Otkrivanje objekata u sustavu 25

4.2. Praćenje objekata u sustavu 26

4.2.1. Podešavanje *agenta* na *Microsoft* poslužiteljima 28

4.2.2. Podešavanje *agenta* na *Linux* distribucijama 31

4.2.3. Podešavanje *SNMP* praćenja 32

4.3. Udaljeno slanje obavijesti o problemima 33

4.3.1. Podešavanje obavijesti pute elektroničke pošte 34

4.3.2. Podešavanje obavijesti putem *SMS-a* 37

4.4. Podešavanje predložaka, grafičko prikazivanje i *proxy* implementacija 39

4.4.1. Uređivanje predložaka 39
4.4.2. Grafičko prikazivanje podataka 42
4.4.3. *Zabbix Proxy* implementacija 44
5. Usporedba sa aplikacijama za *logiranje* 47
5.1. Usporedba s *Microsoft Event Viewer* sustavom 47
5.2. Usporedba s *Linux syslog* servisom 48
Zaključak 49
Popis kratica 51
Popis slika 52
Popis tablica 54
Popis kodova 55
Literatura 57

1. Uvod

Danas poslovne organizacije ovise o pozadinskom informacijskom sustavu pa se može reći da su sustavi okosnica svakog srednjeg i većeg poslovanja. Generiraju veliku korisnost u poslovanjima i doprinose ukupnoj produktivnosti i svrsi poslovanja kao cjeline. Isto tako sustavi teže entropiji i pitanje je vremena kada će biti potrebna intervencija administratora.

2. Pravila, komponente i preporuke

Zabbix rješenje otvorenog koda (engl. *open-source*) kojem je glavna zadaća prikupljanje i spremanje podataka u svrhu uspoređivanja s definiranim pragom (engl. *trashold*). Ako prikupljena informacija prelazi okidač (engl. *trigger*), onda poslužitelj generira upozorenje. Podaci se spremaju u bazu kako bi se mogli prezentirati grafički za željeni vremenski interval. Prvu verziju *Zabbix* poslužitelja (engl. s*erver*) kreirao je Alexei Vladishev 1998., ista verzija izašla je u javnost 2001. godine. Od prve verzije postoje specifične smjernice prema kojima je izrađen sljedeći sustav za centralno praćenje:

- svim pravilima (engl. *rule*), okidačima (engl. *trigger*) i upozorenjima (engl. *alert*) upravlja isključivo poslužitelj
- konfiguracijske radnje obavljaju se *Zabbix* grafičkim *web-sučeljem*
- grafičkom korisničkom *web-sučelju* (engl. *graphical user interface* skraćeno GUI), koje je bazirano u PHP[1] jeziku, pristupa se preko *web-preglednika* te se u pozadini koristi *web-poslužitelj*
- svi podatci spremaju se u relacijsku bazu podataka (*MySQL, MariaDB, PostgreSQL, SQLite, Oracle* ili *IBM DB2*)
- *Zabbix* je pisan u C programskom jeziku zbog što manjeg opterećenja (engl. *footprint*) na pozadinsku infrastrukturu.

2.1. Opis problema

Kako bi se vrijeme prepoznavanja problema i reakcija na potonje ubrzalo, postoje nekolicina proizvoda na tržištu kojima je cilj otkriti probleme. Najgore što se može dogoditi jest nedostupnost nekog servisa koji donosi prihod organizaciji ili je neophodan za rad osoblja, a još gore je kada smo prisiljeni problemu pristupiti reaktivno. U takvim slučajevima potrebno je specifično promatrati pojedine komponente servisa i složiti sustav koji će upozoravati na probleme ili odskakanja od uobičajenog ponašanja. Svaki od servisa u IT poslovanju ima svoju korisnost. Neki izravno doprinose povećanju prihoda organizacijama kao npr. *online* trgovina. Drugi optimiziraju poslovne procese i komunikaciju. Mogu biti različiti servisi za upravljanje imovinom, kupcima pa čak i određeni servisi bez kojih nije moguće obavljati svakodnevne zadatke, npr. slanje e-pošte. Takvi servisi jednako su bitni

[1] *Hypertext Preprocessor* programski jezik je namijenjen za programiranje *web* stranica.

iako ih, kada sve radi normalno i nemamo problema, uzimamo zdravo za gotovo. Nedostupnost bilo koje kategorije servisa uzrokuje pad produktivnosti što posljedično uzrokuje financijske gubitke i nezadovoljne korisnike te štetu reputaciji tvrtke. Svako poslovanje minimizira ovaj rizik implementacijom različitih mehanizama za proaktivno djelovanje na pojavu problema. Postoje različiti načini za adresiranje spomenute problematike, ali jedini način koji ima smisla jest centralno sučelje na kojem se može vidjeti trenutno stanje sustava. Na tržištu postoje različiti proizvodi, no uz malo više uloženog vremena moguće je podići funkcionalan sustav koristeći rješenje otvorenog koda *Zabbix*. Također, bitno je da navedeno rješenje može obavijestiti ljude koji mogu prepoznati utjecaj anomalije na korisničke servise te reagirati na vrijeme.

Cilj je minimizirati cijenu vremena nedostupnosti (engl. *downtime*) za sve servise u poslovanju, a tako i za servis koji prati ostale servise. Od izuzetne je vrijednosti koristiti visoko dostupni pristup dizajnu sustava za praćenje. Kada podebljamo robusnost sustava za praćenje, automatski utječemo i na sustav kao cjelinu, zato što smanjujemo mogućnost da će glavni poslužitelj, koji je imao ulogu javljati problematiku na sustavu, prestati s operativnim djelovanjem. Spomenuta implementacija ima smisla i ako se održava tuđi sustav na našim *Zabbix* poslužiteljima. Što je više uređaja u održavanju, povećava se potencijalna mogućnost pada nekog od servisa. U pravilu postoje različite razine odaziva IT usluga i ovisno o definiranoj razini pregovara se cijena održavanja. Ako se ne zadovolji dogovorena razina, onda se plaćaju penali definirani ugovorom.

Tablica 2.1 Razine dostupnosti IT usluga

Dogovorena dostupnost u postotku	Maksimalna godišnja nedostupnost
90%	36.5 dana
99%	3.65 dana
99.9%	8.76 sati
99.99%	52.56 minuta
99.999%	5.26 minuta
99.9999%	31.5 sekundi
99.99999%	3.15 sekundi

2.2. Komponente *Zabbix* servera

Zabbix poslužitelj (engl. *server*) glavna je komponenta zadužena za prikupljanje i primanje podataka iz okoline koju želimo pratiti. Komunicira s *agentima* i centralnom bazom podataka. *Zabbix* glavna aktivna komponenta grafičko je sučelje preko kojeg korisnik pristupa servisu te vidi prikupljene podatke i može obaviti konfiguraciju. Baza podataka glavni je repozitorij za podatke. U navedenu komponentu pišu se i kasnije čitaju sve vrijednosti neophodne za rad poslužitelja. Dobra praksa jest bazu postaviti na poseban *cluster*[2] poslužitelja. Spomenute komponente obavezne su za rad sustava i u slučaju problema s bilo kojom, sustav za praćenje neće raditi ispravno. *Zabbix proxy*[3] je odvojeni poslužitelj koji pomaže uglavnom u prikupljanju podataka te može biti od koristi za praćenje udaljenih lokacije ili lokacija s ograničenim pristupom. Svrha je smanjiti opterećenje centralnog poslužitelja na kojem je instaliran glavni servis. Prikupljeni podatci spremaju se u bazu na *proxy* poslužitelju i onda se tek šalju prema centralnom serveru koji prezentira sve informacije na korisničko sučelje.

2.3. Preporuke za implementaciju

Prilikom implementacije potrebno je napraviti detaljnu pripremu, u suprotnom će biti problema u funkcioniranju servera ili se neće iskoristit čitavi potencijal u smislu korisnosti za organizaciju koja koristi *Zabbix* za praćenje sustava. Potrebno je znati postaviti i konfigurirati komponente *Zabbix* poslužitelja.

Željena je konfiguracija, visoka dostupnost (engl. *high availability*) na razini aplikacije i baze podataka za što će biti potrebno više virtualnih poslužitelja. Nužno je unaprijed pripremiti popis servisa, aplikacija i uređaja koji će se dodavati u nadzor. Bitno je imati navedeni dio detaljno raspisan kako bismo lakše i efikasnije mogli definirati načine za praćenje te pokriti sve bitne stavke sustava. U smislu *Zabbix* poslužitelja svaki objekt (engl. *host*) koji se promatra organizira se u grupu objekata (engl. *host group*) sličnih ili istih komponenti. *Hostovi* mogu biti u više grupa od jednom, npr. *Microsoft Exchange* serveri mogu biti u grupi naziva „*Exchange* poslužitelji“ i u isto vrijeme u grupi „*Microsoft*

[2] Skup dvaju ili više poslužitelja u svrhu kreiranja kontinuirane dostupnosti servisa.
[3]Posrednički poslužitelj (engl. *proxy*) računalo je koje stoji između dvaju servisa ili između klijenta i poslužitelja u svrhu manipulacije tijeka prometa.

poslužitelji“. Ova logička shema praćenja servisa potrebna je i korisna kada je napravi iskusni administrator koji zna ponašanje sustava duže vrijeme (engl. *baseline*).

Također, dobra je praksa pripremiti predloške (engl. *tamplates*) i definirati ispravne pragove za upozorenja što je specifična karakteristika sustava koji se prati. Predlošci se apliciraju na grupu objekata (engl. *host grupe).* Predefinirane razine okidača generiraju puno lažnih uzbuna (engl. *false positiv*) pa ih je potrebno pripremiti za produkcijsko okruženje. Za pojedini predložak možemo urediti vrijednost okidača koji generira upozorenje ili ih u krajnjem slučaju onemogućiti ako generiraju nelegitimne informacije. Navedene je vrijednosti dobro unaprijed znati jer je svaki servis specifičan. Nije cilj da sustav diže uzbunu kada problema nema pa preoptereti administratore e-poštom te posljedično puni bazu podataka nepotrebnim vrijednostima. Također, moguće je uz mnoštvo dostupnih predložaka preuzeti dodatne predloške koju je kreirala zajednica. Svaki od spomenutih predložaka može se preuzeti sa službenog centralnog repozitorija. Tamo postoje i ocjene predložaka, upute za konfiguraciju i povratna informacija korisnika koji su testirali predložak. Postoji i forum zajednice gdje se može pročitati puno informativnih članaka i adresiranih problema.

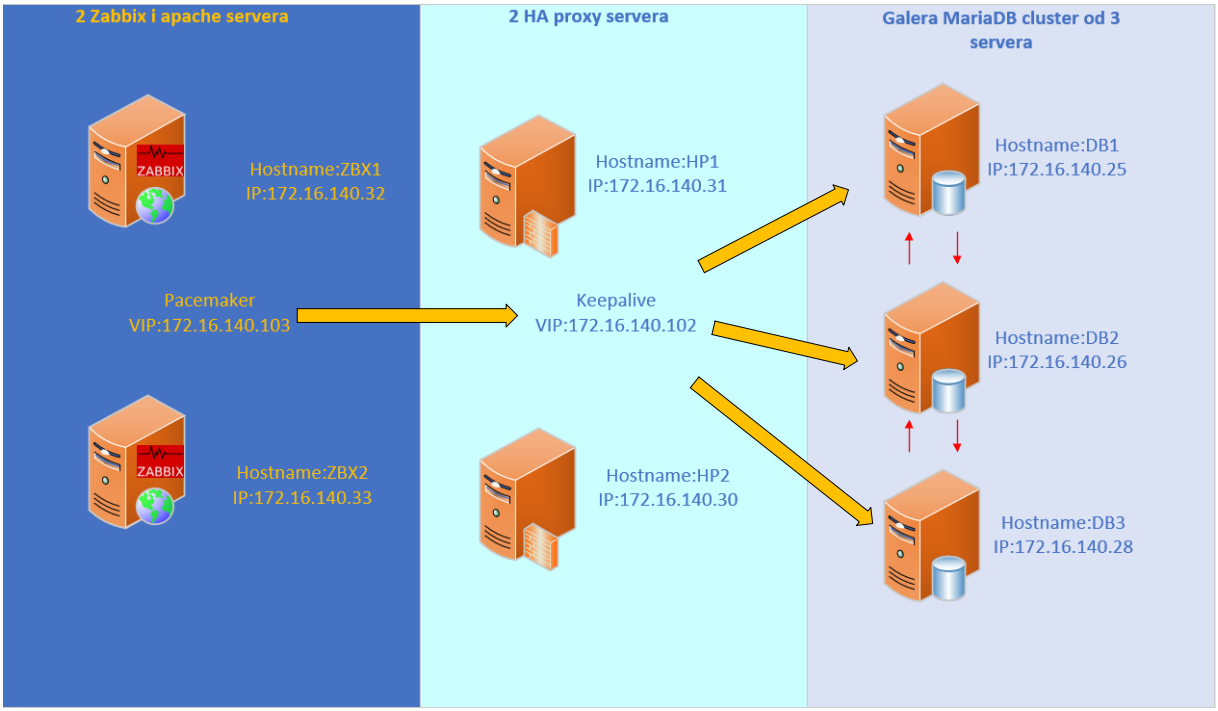
Moramo imati na umu da nije bitno koliko uređaja imamo u nadzoru, već koliko potonji uređaji generiraju prometa po sekundi (engl. *new values per second,* skraćeno nvps). Nije isto motriti jedan server na kojem pratimo jedan servis i poslužitelj za bazu podataka koji će generirati puno više prometa jer je samo praćenje većeg opsega i kompleksnije ovisno o količini predložaka koji su aplicirani na objekte koji se motre. Pratit ćemo servise principom manje je bolje te želimo znati samo ako se stvarno dogodi problem. Svako nepotrebno upozorenje i informaciju onemogućit ćemo ili pobrisati. Nije cilj da alat za praćenje javlja neprekidno netočne informacije jer se tako smanjuje i odaziv na problem, a od velike količine obavijesti neće se vidjeti ona jedna bitna. Dakle, cilj je da sustav generira problem samo onda kada problem postoji, sve ostalo nema smisla i uzalud trošimo resurse.

3. Instalacija visoko dostupnog *Zabbix-servera*

Visoka je dostupnost pristup dizajnu sustava na način da se osigura i garantira odaziv aplikacije ili infrastrukturne implementacije sustava u bilo kojem trenutku izbjegavajući planirano i neplanirano gašenje servisa. Cilj je izbjegavanje ili minimiziranje bilo kakvog oblika nedostupnosti servisa (engl. *downtime)*. Treba imati na umu da vrijeme dostupnosti (engl. *uptime*) ne mora uvijek značiti da je servis dostupan, primjerice može se dogoditi problem na mrežnom uređaju. U tome slučaju naša će visoko dostupna aplikacija i dalje biti pokrenuta, ali korisnici neće moći doći do resursa zbog mrežne nedostupnosti što će se potencijalno moći naplatiti penalima ovisno o dogovorenim ugovornim obvezama usluge (engl. *service level agreement*).

Potrebno je misliti na visoku dostupnost na više različitih razina. Na razini aplikacije moramo osigurati visoku dostupnost svih komplementarnih servisa (centralne aplikacije, baze podataka, servisa za mrežno preusmjeravanje). Fizičke infrastrukture na kojoj je aplikacija instalirana mora se sastojati od više poslužitelja (engl. *hypervisor*). Diskovi na fizičkim poslužiteljima i njihova napajanja moraju biti postavljena tako da se zagarantira minimalan rizik od nedostupnosti. Svi mrežni uređaji moraju biti visoko dostupni. Nužno je svaki aspekt testirati simulirajući produkcijsko okruženje. Izvedba visoke dostupnosti na svim razinama servisa na virtualnim mašinama može se složiti na način kao u nižoj slici (Slika 3.1 Izvedba visoke dostupnosti), ali to je jedan od načina na koji se može konfigurirati. Ideja je da se svaki VM koji ima različitu ulogu nalazi na različitom fizičkom poslužitelju kako bismo se osigurali od nedostupnosti u slučaju kvara poslužitelja, na kojem se nalaze virtualne mašine. Znači, pod pretpostavkom da postoji *cluster* fizičkih poslužitelja, svaki od virtualnih poslužitelja koji spada u jednu od kategorija (glavni servis, *proxy* ili baza podataka) trebao bi biti na različitom fizičkom poslužitelju (engl. *node*) kako bismo minimizirali rizik. Također, u produkcijskom okruženju konfigurirale bi se dodatne mjere zaštite podataka na način da bi se koristile tehnologije repliciranja virtualnih mašina na nekoj drugoj lokaciji, gdje se nalaze poslužitelji namijenjeni za preuzimanje uloge, ako čitava primarna strana otkaže. Usto, osigurala bi se sigurnosna kopija (engl. *backup*) na razini poslužitelja i baze podataka da možemo vratiti u prijašnje stanje ako se dogodi nepopravljiva komprimiranost servisa. Ako želimo postići robustan sustav, onda ćemo zakupiti i usluge

oporavka od katastrofe (engl. *disaster recovery*) od nekog ponuđača takvih usluga u oblaku (engl. *cloud computing*), npr. *Azure*[4] ili *Amazon Web Servises* (AWS[5]).



Slika 3.1 Izvedba visoke dostupnosti

3.1. Predradnje na Centos serveru

Predradnje će se obaviti na svih sedam *CentOS* servera. Moguća je instalacija operativnog sustava u bilo kojoj inačici instalacije, u ovom slučaju odabrana je verzija bez grafičkog sučelja (engl. *compute node*) iz razloga što će se većina administracije odvijati kroz *web-sučelje*. Po dva poslužitelja bit će namijenjena za *Zabbix*, *Apache* i *HAproxy*, a baza podataka bit će smještena u *cluster* od tri poslužitelja. Nakon instalacije operativnog sustava konfigurirat ćemo mrežne kartice tako da svi poslužitelji budu u istoj mreži sa statičkim IP *adresama*. Preimenovat ćemo sve poslužitelje u željenu nomenklaturu te izvršiti ažuriranja svih trenutno instaliranih paketa `yum update -y && yum upgrade -y`. Postavit ćemo ispravnu vremensku zonu na svim poslužiteljima `timedatectl set-timezone Europe/Zagreb`. Uredit ćemo `/etc/hosts` datoteku te svim serverima dodati sljedeće zapise, kako bismo omogućili unutarnje raspoznavanje po imenu (engl. *Domain Name*

[4] Azure je usluga računarstva u oblaku tvrtke Microsoft koja je puštena u produkciju u veljači 2010. godine.
[5] AWS je usluga računarstva u oblaku tvrtke Amazon koja je puštena u produkciju u ožujku 2006. godine.

System resolving). Konfiguracijom *hosts* datoteke izbjegli smo potencijalne probleme međusobnog prepoznavanja poslužitelja prilikom pokretanja servisa koji kao izvor za prepoznavanje koriste ime računala.

```
172.16.140.25 db1
172.16.140.26 db2
172.16.140.28 db3
172.16.140.32 zbx1
172.16.140.33 zbx2
172.16.140.30 hp1
172.16.140.31 hp2
```

Kod 3.1 Izgled *hosts* datoteke na svakom virtualnom poslužitelju

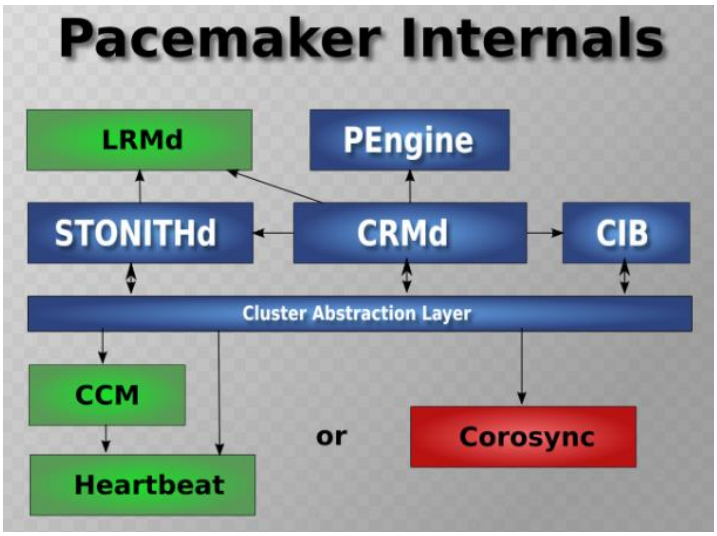
3.2. Visoke dostupnosti na razini *Zabbix* i *web-poslužitelja*

Kako ne bismo ovisili o jednom poslužitelju na kojem su smještene glavne komponente potrebne za funkcioniranje sustava instalirat ćemo *cluster* servis koji služi za održavanje neprekidne dostupnosti. Jedan će poslužitelj biti aktivan, a drugi će preuzimati ulogu ako prvi otkaže (engl. *active passive failover*). Servis naziv *Pacemaker* ima ulogu maksimiziranja dostupnosti svih resursa u *clustera* tako da reagira ako se dogodi greška na nekom od resursa (servisa) ili poslužitelja (engl. *node*). Između dvaju poslužitelja bit će virtualna IP *adresa* kojom će se pristupati na glavno sučelje za administraciju i konfiguraciju praćenja, odnosno komponentu *Zabbix web-sučelje*. Za navedenu implementaciju potrebno je na oba poslužitelja, koji će biti u *clusteru,* imati lokalnog korisnika koji će imati isto korisničko ime i zaporku. Kreirani korisnik imat će svrhu autentifikacije i autorizacije *pacemaker clustera*. Implementacija sadrži odvojene pozadinske servise (engl. *daemon),* odnosno vrstu programa na *Linux* operativnim sustavima koji radi u pozadini, a ne pod izravnim nadzorom korisnika.

Cluster ima svoju centralnu bazu u kojoj zapisuje svoje trenutno stanje. Glavna baza informacija u *clusteru* (engl. *Cluster Information Base*, skraćeno CIB) koristi *EXtensible Markup Language* (skraćeno, XML[6]) kako bi distribuiralo i sinkroniziralo trenutno stanje konfiguracije *clustera* i resursa. Sadržaj CIB-a automatski se replicira i konzistentno je kroz cijeli *cluster*. *PEngine* je komponenta koja koristi vrijednosti iz CIB-a za izračun i postizanje idealnog stanja u *clusteru*. Instrukcije se pohranjuju na poslužitelj koji ima ulogu glavnog kontrolora (engl. *Designated Controller* skraćeno DC), odnosno ima svrhu donošenja odluka

[6] XML je jezik za označavanje podataka.

u *clusteru*. *Pacemaker* centralizira sve odluke na jedan glavni poslužitelj (engl. *master node*) koji ima ulogu upravljača *cluster* resursima (engl. *Cluster Resource Management Daemon*, skraćeno CRMd) sve akcije, promjene, a premještanja unutar *clustera* usmjerene su kroz spomenutu komponentu. Svaki poslužitelj u *clusteru* ima svoj upravljač lokalnim resursima (engl. *Local Resource Management Daemon*, skraćeno LRMd). Djeluje kao sučelje između pojedinih servisa koji su lokalno na poslužitelju i upravljača *cluster* resursima, tj. CRMd-a. Upravljač lokalnim resursima šalje naredbe upravljača *cluster* resursima kao što su pokretanje i zaustavljanje i prenošenje informacija o statusu resursa. *Shoot the Other Node in the Head (*skraćeno, STONITH) djeluje kao izvor *clustera*. STONITH je zapisan u CIB-u i može se nadzirati kao uobičajeni resurs *clustera* koji služi za izbacivanje pojedinog člana iz *clustera* ako ne radi ispravno. *Corosync* komponenta služi kao temelj komunikacije članstva *clustera* i njegovih pojedinih članova.



Slika 3.2 Arhitektura *pacemakera[12]*

3.2.1. Konfiguracija visoke dostupnosti na razini aplikacije i *web-servera*

Konfiguracija se odnosi samo na poslužitelje naziva zbx1 i zbx2, na njima će se instalirati servisi poslužiteljski *Zabbix-server* servis i *web-poslužitelj* naziva *Apache* kao i servise visoke dostupnosti prije spomenuti *pacemaker* i *corosync*. Većinu koraka potrebno je obaviti na oba poslužitelja, a napomene o suprotnome bit će naglašene.

Potrebno je preuzeti paket i instalirati paket httpd koji će imati ulogu pokretanja i održavanja *web-sučelja Zabbix* poslužitelja koje je u suštini *web-stranica*.

```
yum -y install httpd
```

Kako bi poslužitelj radio ispravno, potrebno je postojeću *php* konfiguraciju proširiti ekstenzijama.

```
yum -y install php-bcmathphp php-pear php-cgiphp-xml
php-mysql php-common php-mbstring php-snmp php-gd php-gettext
```

Kako bi se moglo doći do instalacije poslužitelja, nužno je dodati repozitorij. Verzija 4.0 zadnja je stabilna verzija u vrijeme pisanja ovog rada.

```
rpm -ivh
https://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-
release-4.0-1.el7.noarch.rpm
```

Kada je repozitorij dodan *CentOS*, poslužitelj može *yum* komandom doći do svih paketa koji su potrebni za rad poslužitelja za praćenje sustava.

```
yum install -y zabbix-server-mysql zabbix-web-mysql zabbix-
agent zabbix-get
```

Kako bi se uredno prošli svi preduvjeti instalacije, postavit ćemo ispravne vrijednosti vremenske zone za *web-poslužitelj*.

```
vi /etc/httpd/conf.d/zabbix.conf
php_valuedate.timezone Europe/Zagreb
```

Sljedeći je korak uređivanje konfiguracijske datoteke za *Zabbix* poslužitelja komandom `vim /etc/zabbix/zabbix_server.conf`. Za `SouceIP` i `ListenIP` definirat će se virtualna adresa *pacemaker clustera* koju ćemo konfigurirati u jednom od sljedećih koraka. Za `DBHost` unaprijed ćemo postaviti virtualnu *IP adresu clustera* baza podataka.

```
SourceIP=172.16.140.103
ListenIP=172.16.140.103
DBHost=172.16.140.102
DBName=zabbix
DBUser=zabbix
DBPassword=centos
```

Kod 3.2 Parametri za promjenu u konfiguracijskoj datoteci

Moramo popustiti sve potrebne servise kroz vatrozid (engl. *firewall*) kako bismo mogli omogućiti nesmetano spajanje na aplikaciju. Popustit ćemo *portove* za *Zabbix* poslužitelj, *web-server* i visoku dostupnost te nakon toga osvježiti i primijeniti konfiguraciju lokalnog vatrozida.

```
firewall-cmd --add-service={http,https} --permanent
firewall-cmd --add-port={10051/tcp,10050/tcp} --permanent
firewall-cmd --permanent --add-service=high-availability
firewall-cmd --add-service=high-availability
firewall-cmd-reload
```

Kod 3.3 Propuštanje servisa kroz vatrozid

Na svakom od poslužitelja preuzet ćemo i instalirati pakete za instalaciju *clustera*, a zatim ćemo kreirati korisnika koji će se koristiti za autorizaciju servisa na zbx1 i zbx2 poslužiteljima.

```
yum install pacemaker pcs -y
passwd hacluster
```

Pokrenut ćemo instalirane *cluster* servis i postavit automatsku aktivaciju prilikom paljenja operacijskog sustava.

```
systemctl start pcsd.service
systemctl enable pcsd.service
```

Ostatak konfiguracije odrađivat ćemo na poslužitelju na kojem će se inicijalno pokrenuti *cluster* servis, a to će biti zbx1. Prvo je potrebno obaviti predradnju autorizacije zbx1 i zbx2 poslužitelja za pacemaker servis.

```
[root@zbx1 ~]pcs cluster auth zbx1 zbx2
```

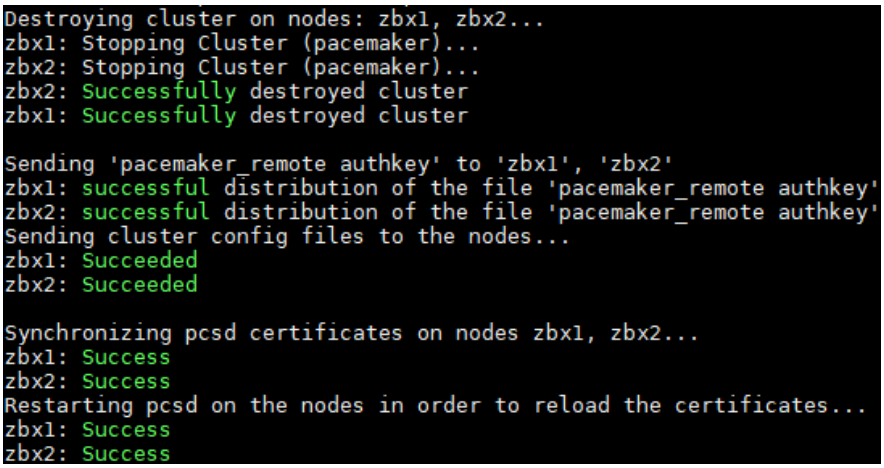
Očekivani rezultat ja status autoriziran kao u ispisu (Kod 3.4 Uspješna autorizacija pcs servisa između poslužitelja). Ako se pojave problemi na ovom koraku, potrebno je provjeriti konfiguraciju `/etc/hosts` datoteke i pravila vatrozida.

```
zbx1: Authorized
zbx2: Authorized
```

Kod 3.4 Uspješna autorizacija pcs servisa između poslužitelja

Ukoliko je autorizacija prošla uspješno, utoliko možemo početi s konfiguracijom *pacemaker* servisa. Prvi korak jest kreiranje *clustera* proizvoljnog naziva koji se sastoji od prije autoriziranih poslužitelja. Kreirat ćemo *cluster* naziva *zabbix_claster* koji se sastoji od članova zbx1 i zbx2. Poslužitelji će obrisati prijašnju konfiguraciju *cluste*ra ako postoji te izmijeniti ključeve i certifikate.

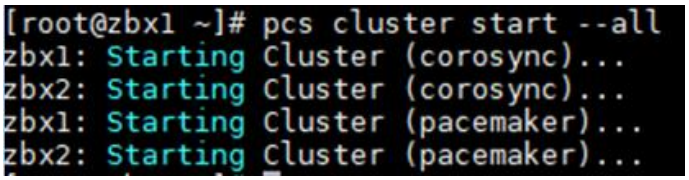
```
[root@zbx1 ~]pcs cluster setup –name zabbix_cluster zbx1 zbx2
```



Slika 3.3 Očekivani ispis nakon pokretanja *clustera*

Kada je inicijalno postavljeno prošlo s uspješnim statusom, onda možemo pokrenuti novokreirani *cluster*, a to će pokrenuti servise koji su za to potrebni na oba člana.

```
[root@zbx1 ~]pcs cluster start --all
```



Slika 3.4 Pokretanje servisa nakon uspješnog *setupa*

Sada, kad su servisi pokrenuti, potrebno omogućiti njihovo garantirano i automatsko pokretanje prilikom pokretanja operacijskog sustava.

```
systemctl enable corosync.service
systemctl enable pacemaker.service
```

Cluster će se brinuti za stanje servisa, a mi ćemo se pobrinuti ispraviti pogrešku ako se dogodi na jednom od poslužitelja. Ne želimo da *cluster* izbacuje članove automatski, poglavito zbog činjenice jer je konfiguriran c*luster* koji se sastoji od dvaju poslužitelja. Ako nam iz nekog razloga servis izbaci jednog, onda nećemo imati c*luster*, već samo jedan poslužitelj na kojem je servis. Nećemo koristiti prije spomenuto STONITH svojstvo pa ćemo ga i isključiti.

```
pcs property set stonith-enabled=false
```

Također, u *clusteru* se članovi moraju dogovoriti na kojim će poslužiteljima biti pokrenuti servis i gdje će se nalaziti pojedini resursi. Ako u *clusteru* ima neparan broj članova, onda je za odluke potreban kvorum, tj. minimalan broj članova kako bi se postigao broj koji je veći od pola. Potonji postupak isto će se isključiti jer nema koristi u našoj implementaciji.

```
pcsproperty set no-quorum-policy=ignore
```

Cluster je podešen i sada možemo kreirati resurse u njemu. Prvi resurs koji je ujedno potreban za funkcioniranje ostalih je virtulana *IP adresa* (skraćeno, *VIP*) koja je potrebna da se naši članovi *clustera* mogu predstavljati u njezino ime. Navedenu adresu definirali smo unaprijed u konfiguracijskoj datoteci `zabbix_server.conf` pod parametrima `SourceIP i ListenIP`, a kreirat ćemo je kao funkcionalni resurs *clustera*. Adresa će biti 172.16.140.103/24 kao što je prije definirano.

```
pcs resource create cluster_vip ocf:heartbeat:IPaddr2
ip=172.16.140.103cidr_netmask=24 op monitor interval=15s
```

Na sličnu način dodaje se i resurs *zabbix-server* servisa. Kreira se resurs naziva *zabbix_server* koji će pratiti *systemd* serveri *zabbix-server*.

```
pcs resource create zabbix_server systemd:
zabbix-server op monitor interval=5s
```

Kao resurs *clustera* moramo dodati i još jedan krucijalni servis, a to je *httpd,* tj. *Apache web-poslužitelj* na kojem se vrti naše *web-sučelje.*

```
pcs resource create httpd systemd:httpd op monitor interval=5s
```

Kako bismo olakšali administraciju i konfiguraciju *cluster* servisa, grupirat ćemo resurse za servise *httpd* i *zabbix-server* u jednu grupu resursa koja će se zvati *zabbix_httpd.*

```
pcs resource group add zabbix_httpd zabbix_server httpd
```

Kako bismo osigurali da se svi resursi budu pokrenuti na jednom od dostupnih članova (engl. *node*), kreirat ćemo skup ograničenja. Osigurat ćemo da će resursi biti pokrenuti na pojedinom članu *clustera* te da se, ako se dogodi da ni jedan poslužitelj nije dostupan, onda ni servisi neće pokušavati pokrenuti.

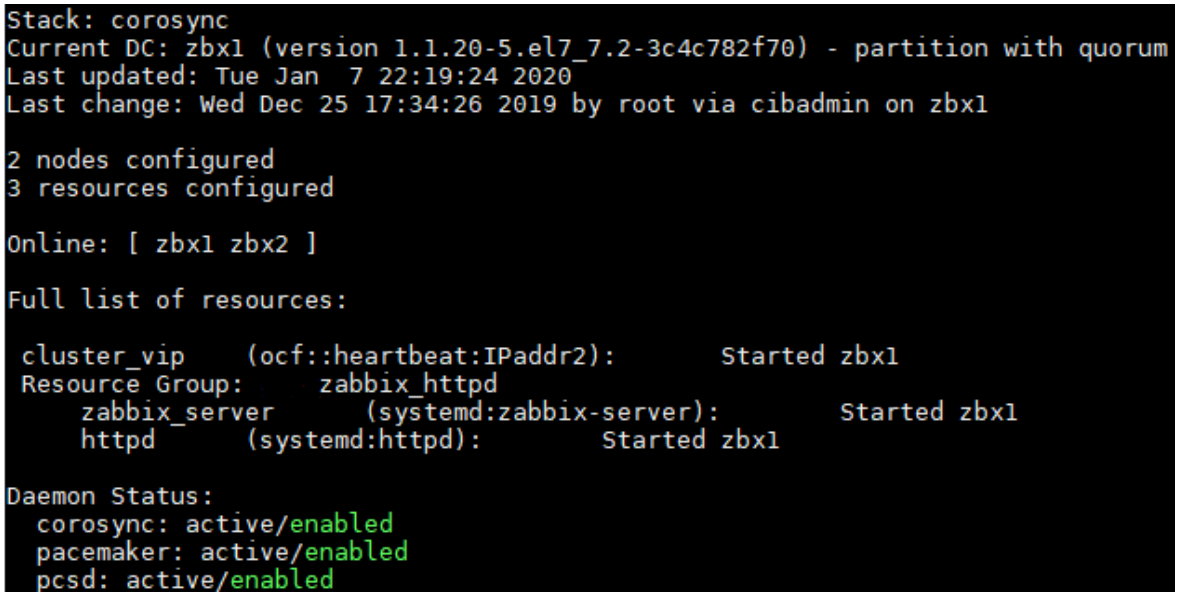
```
pcs constraint colocation add zabbix_httpd cluster_vip INFINITY
```

Također, za *cluster* je bitno da se prvo pokrene virtualna IP *adresa* zato što je ona uvjet za pokretanje servisa iz grupe *zabbix_httpd*. Bez VIP *adrese* nećemo moći pristupiti aplikaciji pa je zato potrebno podesiti određeni poredak paljenja servisa tj. dat ćemo prioritet paljenju prvenstveno VIP *adresi*, a potom grupi resursa *zabbix_httpd*.

```
pcs constraint order cluster_vip then zabbix_httpd
```

Ako je sve posloženo kako treba i ne postoje problemi sa *Security-Enhanced* (skraćeno, SE) *Linuxom* i vatrozid (engl. *firewall*), komanda provjere `pcs status` trebala bi imati ispis kao na slici niže (Slika 3.5 Funkcionalno stanje *pc sclustera* s podešenim resursima). Ako status ispisuje greške, potragu za rješenjem uvijek je dobro početi pregledom *log* zapisa o servisu

za koji se greška javlja, primjerice ako je problem s pokretanjem *zabbix-server* servisa, onda provjeravamo zapise o događajima naredbom `tail -l /var/log/zabbix/zabbix_server.log`.



Slika 3.5 Funkcionalno stanje *pc sclustera* s podešenim resursima

Trenutno pcs servis radi kako treba. Kako bismo izbjegli probleme prilikom spajanja na udaljeni *cluster* baza podataka obavit će se propuštanja kroz *SELinux*[7]. Moramo dozvoliti *Zabbix* servisu spajanje na mrežu te spajanje na udaljenu bazu podataka. Također, obavit ćemo propuštanje servisa prema *audit logovima*.

```
setsebool -P zabbix_can_network 1
setsebool -P httpd_can_network_connect_db 1
setsebool -P httpd_can_connect_zabbix 1
cat /var/log/audit/audit.log | grepzab | audit2allow -M zabbix-server
semodule -i zabbix-server.pp
```

Kod 3.5 Propuštanje servisa kroz *SELinux*

3.3. Visoka dostupnost na razini baze podataka

Potrebno je omogućiti *Zabbix* serverima da po *portu* 3306 mogu pristupiti bazi podataka. Na razini servisa *zabbix-server* i *apache* konfigurirana je visoka dostupnost (engl. *high availabilty*), ali kako bi prijašnje radnje imale smisla, isto je potrebno napraviti i na razini baze podataka kako bi se omogućila koherentna visoka dostupnost za ove komplementarne servise. Moramo osigurati neovisnost o jednom VM-u za bazu podataka,

[7]Sigurnosni modul na Linux distribucijama koji pruža mehanizme za kontrolu pristupa

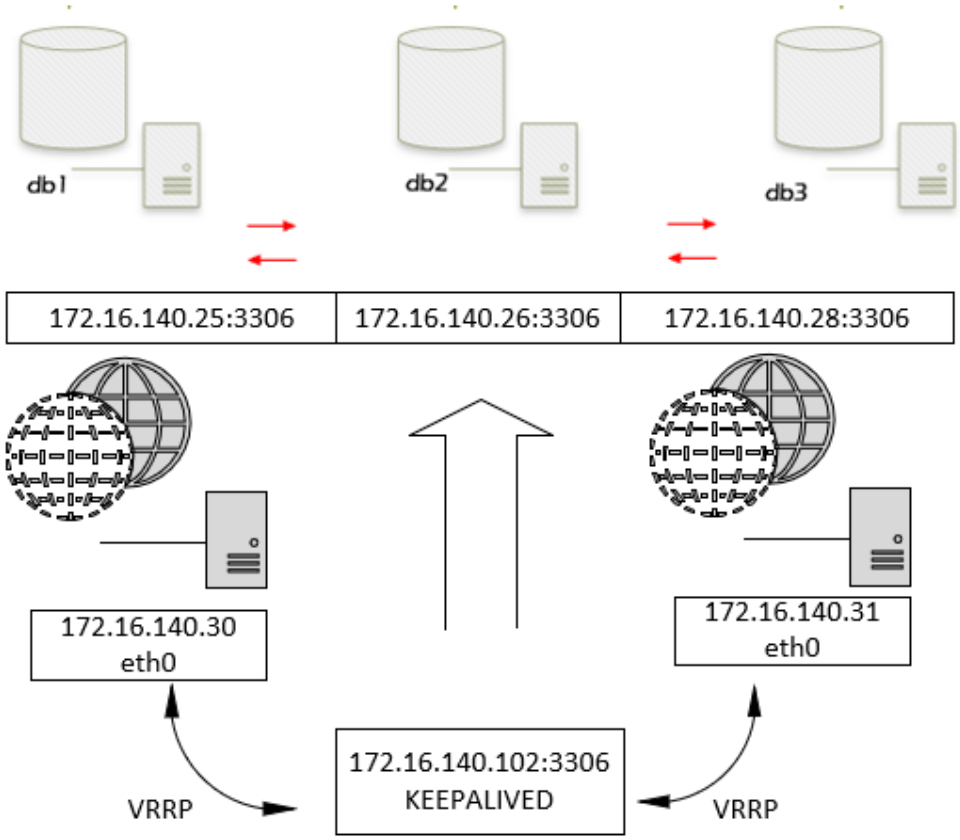
zato što je uzaludno imati odvojeni aplikativni *cluster* ako dostupnost baza nije osigurana. *Zabbix* se spaja na bazu i neće biti puno koristi od *pacemaker* servisa ako se dogodi nedostupnost na razini poslužitelja na kojem se nalazi baza podataka. Servis *Galera cluster* način je konfiguracija dvaju ili više poslužitelja za sinkronizirani način replikacije podataka. Kroz *Galera* grupnu komunikaciju svi poslužitelji će imati konzistentno stanje baze podataka. Korisnici ne moraju znati na koji od servera pišu ili čitaju.



Slika 3.6 *Galera* grupna komunikacija

Međutim, kako bi se tri servera predstavljali ispred jedne IP adrese, potrebno je podignuti *HAproxy* koji će preusmjeravati promet i na njima *Keepalive* servis koji će podići *VIP* za spajanje na *Galera cluster*. *HAproxy* je popularno rješenje za ujednačavanje opterećenja (engl. *load balancing*) i često se koristi za distribuciju prometa na više servera. U ovoj implementaciji podići ćemo dva poslužitelja u tu svrhu kako bismo izbjegli jednu točku pogreške (engl. *single point of failure*) i olakšali radnje održavanja. Na *proxy serverima* postaviti ćemo *keepalived* servis koji će služiti kao virtualni IP prema *serverima* na kojima će se nalaziti baza podataka za aplikaciju. Potonje je potrebno napraviti zbog načina spajanja *Zabbix servera* na bazu.

Naime, prilikom konfiguracije konekcije s bazom navodi se isključivo jedna IP *adresa*. Iz tog razloga potrebno je da se iza jedne adrese nalazi više *servera* jer inače *cluster* baza podataka neće imati smisla. Primjerice, ako definiramo konekciju samo prema db1 *serveru* i isti taj *server* otkaže, imat ćemo problem, a ako se tri *servera* javljaju pod istom IP *adresom*, taj ćemo rizik minimizirati. Malo je vjerojatno da će sva tri poslužitelja imati problem. *Keepalive software* za usmjeravanje koristit će se kao VIP *adresa* koja će biti uvijek dostupan na jednom od *proxy* poslužitelja, a preko iste adrese *Zabbix server* čitat će i pisati u bazu podataka.



Slika 3.7 Implementacija *Galera cluster*a zajedno s *keepalived* servisom na *HAproxy* poslužiteljima

3.3.1. Konfiguracija *Galera* servisa visoke dostupnosti na razini baze podataka

Podignut ćemo tri VM-a s *CentOS* 7 operacijskim sustavom. Nazvat ćemo ih db1, db2 i db3. Na njima ćemo odraditi inicijalnu konfiguraciju kao u predradnjama pa ćemo instalirati zadnju stabilnu verziju *MariaDB* poslužiteljskog servisa. Na svakom od poslužitelja potrebno je obaviti identičnu konfiguraciju. Na lokaciji `/etc/yum.repos.d/` dodat ćemo repozitorij naredbom `nano /etc/yum.repos.d/mdb.repo`. Kopirat ćemo zadnju stabilnu verziju sa službene stranice. U vrijeme pisanja rada zadnja stabilna verzija jest 10.4.

```
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.4/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Kod 3.5 zapis za repozitorij *MariaDB* poslužitelja

Kad je dodan repozitorij, možemo skinuti i instalirati *MariaDB* servis. Nakon instalacije pokrenut ćemo servis na svim *serverima*.

```
yum install MariaDB-server MariaDB-client -y
```

Kada je servis instaliran na poslužitelj, onda ga možemo pokrenuti i omogućiti automatsko pokretanje.

```
systemctl start mariadb
systemctl enable mariadb
```

Pokrenuvši servis možemo odraditi inicijalnu konfiguraciju poslužitelja baze podataka .

```
mysql_secure_installation
```

Bitno je da u inicijalnom instalacijom definiramo osnovne parametre jer je brži način, ali isto je moguće odraditi i ručno komandom za svako postavljeno pitanje.

```
Change root password? Y
Remove anonymous users? Y
Disallowroot login remotely? Y
Remove test database and access to it? Y
Reload privilege tables now? Y
```

Kod 3.6 Instalacija *MariaDB* poslužitelja koristeći *shell* skriptu

Na lokaciji je potrebno kreirati konfiguracijsku datoteku u kojoj će se definirati parametri kao što su ime *cluste*r, baza podataka i popisa članova u njemu:

```
nano /etc/my.cnf.d/galera.cnf
```

Konfiguracijsku datoteku može se ispuniti kao u kodu niže, jedino što je potrebno mijenjati jest IP *adresa* i ime poslužitelja na kojem se konfigurira. Uz postavke poslužitelja baze podataka u konfiguraciji je potrebno definirati ime *cluste*r i IP *adrese* svih poslužitelja, taj dio se ne mijenja.

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
default_storage_engine=MyISAM
innodb_strict_mode = 0
max_allowed_packet=32M

#GaleraProviderConfiguration
wsrep_on=ON
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

#GaleraClusterConfiguration
wsrep_cluster_name="mdb_cluster"
wsrep_cluster_address="gcomm://172.16.140.25,172.16.140.26,172.16.1
40.28"

#GaleraSynchronizationConfiguration
wsrep_sst_method=rsync
```

```
# GaleraNodeConfiguration, ovaj dio se mijenja od servera do
servera. Mijenja se adresa i hostname
wsrep_node_address="172.16.140.25"
wsrep_node_name="db1"
```

Kod 3.7 Izgled konfiguracijske datoteke galera *cluste*r na db1

Ono što se mora promijeniti prilikom konfiguracija različitih poslužitelja parametri su na slici niže (Slika 3.8 Izgled konfiguracijske datoteke) je `wsrep_node_address` tj. IP adresa i `wsrep_node_name`, tj. ime virtualnog računala.

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
default_storage_engine=MyISAM
innodb_strict_mode = 0
max_allowed_packet=32M


# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="mdb_cluster"
wsrep_cluster_address="gcomm://172.16.140.25,172.16.140.26,172.16.140.28"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="172.16.140.25"
wsrep_node_name="db1"
```

Slika 3.8 Izgled konfiguracijske datoteke *Galera clustera*

Na svim poslužiteljima stopirat ćemo *MariaDB* servis, a nakon toga pokrenuti *Galera cluster* na prvom poslužitelju, tj. na db1. Kada se *Galera cluster* inicijalizira, ponovno ćemo pokrenuti *MariaDB* servis na preostalim *serverima*, tj. db2 i db3. Komandnom linijom potrebno je koristiti `systemctl` naredbu te `galera_new_cluster` ovisno o poslužitelju na kojem se pokreće.

```
[root@db1 ~]#systemctl stop mariadb
[root@db2 ~]#systemctl stop mariadb
[root@db3 ~]#systemctl stop mariadb
[root@db1 ~]#galera_new_cluster
[root@db2 ~]#systemctl startmariadb
[root@db3 ~]#systemctl startmariadb
```

Kod 3.8 Redoslijed pokretanja *MariaDB* servisa i inicijalno pokretanja *clustera*

Također, kako bi se izbjegli problemi, potrebno je odraditi propuštanje kroz *SELinux*. Kreirat ćemo jednu bazu podataka u kojoj ćemo unijeti tablicu i vrijednosti pa ćemo prema generiranim *logovima* obaviti propuštanja koristeći `audit2allow` naredbu.

```
#kreirat ćemo bazu podataka dediciranu za selinux propuštanja
mysql -u root -p -e 'CREATE DATABASE selinux;'

#u istoj bazi ćemo kreirati tablicu za selinux politiku
CREATE TABLE selinux.selinux_policy (id INT NOT NULL
AUTO_INCREMENT,PRIMARY KEY(id));

#novoj tablici ćemo dodati vrijednosti kako bi se pojavili logovi
INSERT INTO selinux.selinux_policy VALUES ();'
mysql -u root -p -e 'INSERT INTO selinux.selinux_policy VALUES ();'

#temeljem logova koji su generirani unosom podataka prikupit ćemo
logove
grepmysql /var/log/audit/audit.log | audit2allow -M Galera

#popušanje prema prikupljenim podacima
semodule -i Galera.pp
```

Kod 3.9 Propuštanje *MariaDB clustera* kroz *SELinux*

```
[root@db1 ~]# grep mysql /var/log/audit/audit.log | sudo audit2allow -M Galera
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i Galera.pp
```

Slika 3.9 Propuštanje *Galera cluster* kroz *SELinux*

Nakon što je kreiran *Galera cluster* te obavljena propuštanja, potrebno je provjeriti funkcionalnosti baze podataka. Na proizvoljnom članu *clustera* kreirat ćemo testnu bazu podataka u kojoj ćemo zapisati vrijednosti u neku tablicu. Ista baza mora biti odmah vidljiva na drugim serverima te mora biti moguće i uređivanje iste baze. Potonje možemo testirati dodavanjem vrijednosti u tablicu baze koju smo kreirali na drugim poslužiteljima.

```
#ulazak u mysql servershell
[root@db1 ~]# mysql -u root -p

#kreiranje baze podataka za testiranje
MariaDB [(none)]>Create database test_replikacije;

#Kreiranje tablice na nekom od drugom serveru
MariaDB [(none)]> CREATE TABLE test ( id smallint unsigned not null
auto_increment, name varchar(20) notnull, constraint pk_example
primarykey (id) );

#unos podataka u tablicu na trećem serveru
MariaDB [(none)]> INSERT INTO test ( id, name ) VALUES ( null, 'test' );
```

Kod 3.10Testiranje funkcionalnosti

Odrađena je podizanje *MariaDB cluster* koji će koristiti za visoku dostupnost na razini baze podataka. Sljedeći korak jest iskoristiti funkcionalnost za kreiranje baze podataka koja će koristit *zabbix-server* servis. Potrebno je kreirati praznu bazu podataka te korisniku

koji će imati sva prava na bazi kao i mogućnost spajanja sa udaljene VIP *adrese*, tj. adrese koju koristi *pacemaker cluster.*

```
#ulazak u mysqlshell
mysql -u root -p

#kreiranje baze podataka
CREATE DATABASE zabbix CHARACTER SET utf8 COLLATE utf8_bin;

#dodavanja prava na bazu
GRANT ALL PRIVILEGES ON 'zabbix'.* TO zabbix@'%' IDENTIFIED BY 'centos';

#dopuštanje udaljenog spajanja preko virtualne ip adrese;
GRANT ALL ON zabbix.* TO 'zabbix'@'172.16.140.103' IDENTIFIED BY 'centos';

#primjena dodanih prava
FLUSH PRIVILEGES;
```

Kod 3.10 Kreiranje prazne baze podataka i dodavanje prava

U trenutnu praznu bazu podataka potrebno je ubaciti shemu koju koristi *Zabbix* aplikacija, a ta shema nalazi se na oba *Zabbix* poslužitelja, tj. zbx1 i zbx2. Možemo je prebaciti s proizvoljnog poslužitelja koji ima instaliran *zabbix-server* servis.

```
#Prebacivanje sheme zabbix servera na jedan od galera poslužitelja
[root@zabbix01 ~]# scp /usr/share/doc/zabbix-server-mysql-
                  4.0.12/create.sql.gz root@172.16.140.25:/root/
#Ubacivanje inicijalne šeme
[root@db1 ~]# zcat create.sql.gz | mysql -u zabbix -p zabbix
```

Kod 3.11 Dodavanje inicijalne *sheme* za *zabbix* bazu podataka

Na svim *serverima* potrebno je propustiti sučelje za međusobnu komunikaciju između *MariaDB clustera* te za komunikaciju sa *Zabbix* aplikacijom. Sučelje 3306 služi za *MariaDB* servis, a sučelja 4567, 4568 ,4444 koristi *Galera cluster* replikacija.

```
firewall-cmd --add-
port={3306/tcp,4567/tcp,4568/tcp,4444/tcp,4567/udp,10051/tcp,10050/tcp
            } –permanent

firewall-cmd –reload
```

Kod 3.12 Propuštanja kroz vatrozid za *Galera cluster*

3.3.2. Konfiguracija virtualne *keepalive IP adrese* i *HAproxy* servisa

S trenutnom konfiguracijom postoji jedan problem, a to su tri *MariaDB* poslužitelja u *clusteru*. Kao što je prije spomenuto, prilikom spajanja *Zabbix* aplikacije na bazu podataka konfigurira se samo jedna IP *adresa*. Šteta bi bilo postaviti da se aplikacija povezuje na jedan poslužitelj eksplicitno, tu se opet gubi smisao *clustera*. Za razliku od *Pacemakera Galera*, servis nema način da se podigne još jedan resurs koji će imati ulogu VIP adrese. U tu svrhu podići ćemo dodatna dva *CentOS* poslužitelja. Na njima će biti instalirani servisi *Keepalive* i *HAproxy* s kojima će se adresirati ova problematika. Odradit ćemo podizanje VM-ova te inicijalnu konfiguraciju kao u predradnjama. Konfiguracija će se odrađivati na hp1 i hp2 *serverima*.

Na hp1 i hp2 instalirat ćemo potrebne pakete za konfiguraciju visoke dostupnosti. Nakon uspješnog skidanja nastavit ćemo konfiguracijom, ali prvo ćemo napraviti sigurnosnu kopiju predefinirane konfiguracije *keepalived* servisa.

```
Yum install keepalived haproxy -y
cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.bak
```

Kao predradnja za funkcioniranje *HAproxy servera* moramo posložiti *keepalived* servis na oba poslužitelja. Budući da smo pratili praksu kopiranja inicijalne konfiguracije `keepalived.conf` u *file* za sigurnosnu kopiju iste `keepalived.conf.bak`, možemo obrisati čitav sadržaj inicijalne konfiguracijske datoteke. To je najlakše napraviti kroz `vim` uređivač (engl. *editor*) komandom `1,$d` što znači „obriši sve od prvog do zadnjeg reda“. Poslužitelj hp1 bit će u glavnom (engl. *master mode*) poslužitelju, tj. VIP će se uvijek vraćati na njega u slučaju da postane dostupan nakon prvobitne ne dostupnosti (npr. *restart* poslužitelja).

Prvo ćemo urediti konfiguraciju na hp1 `vim/etc/keepalived/keepalived.conf`. Istu datoteku potrebno je urediti i na drugom *serveru* s malim preinakama. Poslužitelj hp2 bit će u pričuvi (engl. *backup mode*) i imat će manji prioritet kako bi servis preferirao hp1 ako je na mreži (engl. *online*). Niže se nalazi potrebna konfiguracijska datoteka s komentarima (Kod 3.13 Izgled *keepalived* konfiguracijske datoteke). Kad se odradi konfiguracija i pokretanje servisa, sve se može provjeriti komandom `ipaddress` na hp1 poslužitelju, ispis mora prikazivati VIP koji je konfiguriran (Slika 3.10 Željeni ispis komande provjere nakon konfiguriranog *keealived* servisa). Također, prilikom ponovnog pokretanja hp1 poslužitelja VIP se treba prebaciti na hp2 poslužitelj te kasnije vratiti na inicijalnu poziciju, kada se hp1 ponovno podigne.

```
vrrp_scriptchk_haproxy {
script "killall -0 haproxy" # provjera haproxy procesa
interval 2 # svake 2 sekunda
weight 2 # dodaje 2 boda ako je ok
}
vrrp_instance VI_1 {
interface eth0 # mrežno sučelje koje se prati
state MASTER # MASTER je na hp1, BACKUP na hp2
virtual_router_id 51
priority 101 # 101 na hp1, 100 na hp2
virtual_ipaddress {
172.16.140.102 # virtualna ip adresa koje će imati ovo virtualno sučelje
}
track_script {
chk_haproxy
}
}
```

Kod 3.13 Izgled *keepalived* konfiguracijske datoteke

Nakon odrađivanja konfiguracije servisa moramo omogućiti automatsko pokretanje servisa prilikom ponovnog pokretanja poslužitelja te pokrenuti servis prvi put.

```
systemctl start keepalived
systemctl enable keepalived
```

```
[root@hp1 ~]# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:9f:8c:f2 brd ff:ff:ff:ff:ff:ff
    inet 172.16.140.30/24 brd 172.16.140.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
    inet 172.16.140.102/32 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::f2d4:70f5:5404:4c9d/64 scope link tentative noprefixroute dadfailed
       valid_lft forever preferred_lft forever
    inet6 fe80::7df:7aac:8dc8:56fc/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:92:33:2c brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:92:33:2c brd ff:ff:ff:ff:ff:ff
```

Slika 3.10 Željeni ispis komande provjere nakon konfiguriranog *keealived* servisa

Odrađena je konfiguracija *Keepalive* servisa, sad je potrebno podignuti *proxy* servis na poslužiteljima hp1 i hp2. *Keepalive* nudi usluge VIP na više poslužitelja, a *HAproxy* će

za tu adresu promet prebacivati prema jednom od triju poslužitelja na kojima je baza podataka. Kako VIP nalazi primarno na hp1 i po potrebi na hp2 poslužitelju, počet ćemo s konfiguracijom na hp1 poslužitelju. Postavke će biti iste na oba poslužitelja, a *keepalived* će određivati na koji će biti aktivni *proxy* prema *MariaDB clusteru*. Počet ćemo slaganje jednako kao i s prijašnjim servisom. Nakon skidanja potrebnih paketa napravit ćemo sigurnosnu kopiju konfiguracijske datoteke i obrisati čitav sadržaj (u `vim` uređivaču `1,$d`) inicijalnog stanja kako bismo mogli u konfiguracijsku datoteku, koju ćemo koristiti, ostaviti samo relevantne stavke, a opet imali mogućnost povratka u prvobitno stanje.

```
yum install haproxy -y
cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
```

Uredit ćemo pripadajuću konfiguracijsku datoteku `Vim /etc/haproxy/haproxy.cfg` na način da ćemo navesti da *HAproxy* sluša na IP *adresi keepalived* servisa te da se iza nalaze tri *MariaDB* poslužitelja koji se međusobno repliciraju.

```
# LoadBalancing for Galera Cluster
Listen galera 172.16.140.102:3306
Balance source
mode tcp
option tcpka
server db1 172.16.140.25:3306 checkweight 1
server db2 172.16.140.26:3306 checkweight 1
server db3 172.16.140.28:3306 checkweight 1
```

Kod 3.14 Izgled *HAproxy* konfiguracijske datoteke

Nakon odrađivanja konfiguracije servisa moramo omogućiti automatsko pokretanje servisa prilikom ponovnog pokretanja poslužitelja te pokrenuti servis inicijalno.

```
Systemctl start haproxy
Systemctl enable haproxy
```

Kako bismo izbjegli potencijalne probleme sa spajanjem propustit će se *HAproxy* spajanje kroz *SELinux* zaštitu.

```
setsebool -P haproxy_connect_any 1
```

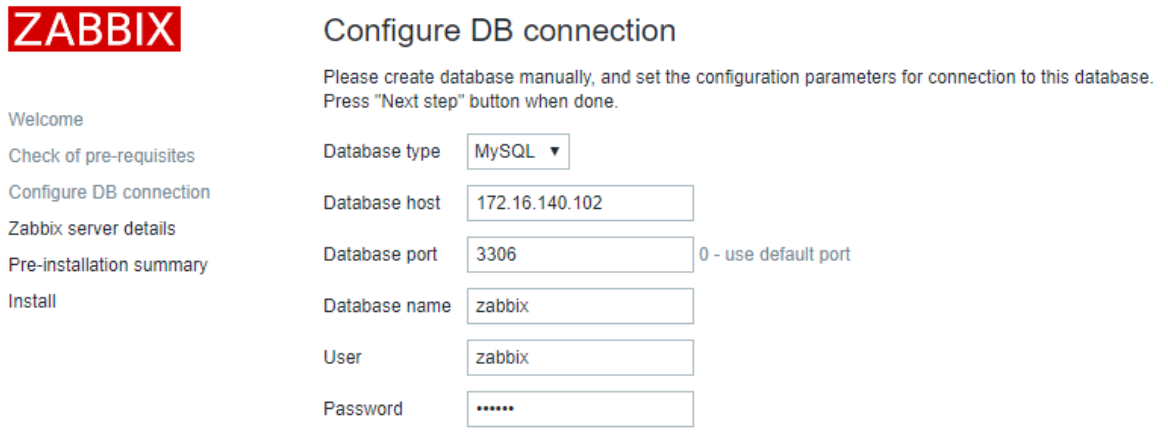
Kako VIP *adresa* neće biti prisutna na oba servera u isto vrijeme, može se inducirati greška na *HAproxy* servisu. Naime, na poslužitelju na kojem se ne nalazi VIP može se pojaviti greška spajanja na virtualnu IP *adresu* i tako posljedično na *cluster* u pozadini. Razlog je što trenutno definirana adresa ne postoji na poslužitelju. Zaobilazno rješenje

ponovno je startati servis u tom slučaju greška nestaje. Kako bismo grešku eliminirali prilikom svakog ponovnog pokretanja poslužitelja, moramo periodički pokrenuti HAproxy servis odmah nakon što se OS podigne. *Crontab*[8] koji će se aktivirati prilikom pokretanja operacijskog sustava i pokretat će skriptu koja aktivira servisa *HAproxy*

```
Crontab -e
@rebootsleep 5 &&  /scripts/startup.sh
#sadržaj skripte startup.sh koja je na lokaciji /scripts/startup.sh
#!/bin/bash
systemctl start haproxy
```

Kod 3.15 Konfiguracija skripte za aktivaciju servisa prilikom ponovnog pokretanja

Sad možemo pristupiti instalaciji *web-sučelju* na adresi 172.16.140.103. Potrebno je definirati povezanost prema bazi koja se nalazi *MariaDB Galera clusteru*. Kroz instalacijski čarobnjak (engl. *wizard*) proći ćemo provjeru preduvjeta te konfiguracije spajanja na bazu podataka preko VIP-a koji se vrti između hp1 i hp2 ovisno koji od njih je na mreži.



Slika 3.11 Spajanje na bazu podataka

Nakon inicijalne konfiguracije *web-sučelja* kopirat ćemo datoteku na drugi poslužitelj (zbx2) kako ne bismo trebali ponovno prolaziti kroz iste korake.

```
scp /etc/zabbix/web/zabbix.conf.php
root@172.16.140.33:/etc/zabbix/web/zabbix.conf.php
```

Kod 3.16 Prebacivanje inicijalne *frontend* konfiguracije na drugi poslužitelj

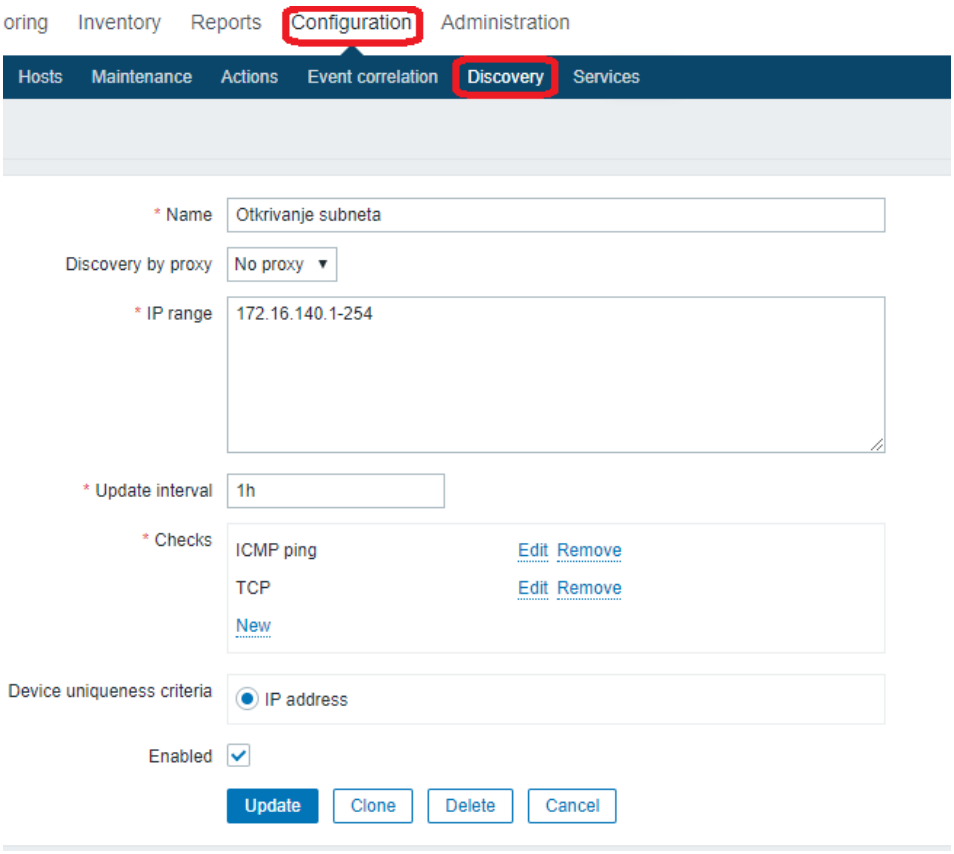
[8] Služi za definiranje periodičkog izvršavanja skripte na *Linux* operacijskom sustavu.

4. Podešavanje funkcionalnosti

U ovom poglavlju pokazat će se kako dodati pojedini poslužitelj u nadzor koristeći *agenta* ili SNMP protokol. Pokazat će se načini za konfiguraciju e-pošte i poruke upozorenja. Podešavat će se predlošci *web-sučeljem* i dovesti visoko dostupni sustav za praćenje u operativno stanje.

4.1. Otkrivanje objekata u sustavu

Zabbix poslužitelj ima mogućnost otkrivanja objekata po određenom protokolu. U konfiguracijskom sučelju možemo definirati različita pravila skeniranja mreže. Mora se definirati mrežni raspon i protokol te pokrenuti otkrivanje (engl. *discovery*). Služi za provjeru odaziva *hosta* u određenom mrežnom rasponu. Pomaže prilikom dodavanja objekata u nadzor. Trenutno, u verziji 4.0 nema intuitivan način za automatizaciju dodavanja objekata u nadzor i istovremenu instalaciju agenta.



Slika 4.1 Otkrivanje *hostova* u *subnetu*

4.2. Praćenje objekata u sustavu

Prikupljanje podataka o trenutnim stanjima uređaja koji se promatraju moguće je postaviti na više načina. Svaki način promatranja konfigurira se kao sučelje koje moramo postaviti za pojedini objekt. ZBX se odnosi na praćenje preko *agenta*:

- *SNMP* je praćenje istoimenim protokolom
- *JMX* se odnosi na praćenje *Java* aplikacije
- *IPMI* se koristi za praćenje fizičkih servera koji imaju integrirano sučelje za dodatne funkcionalnosti i za praćenje pojedinih komponenti.

Kao što je prije spomenuto *hostovi* u smislu *Zabbix* poslužitelja jedinke su koje promatramo kao fizičke ili virtualne servere, mrežni ili bilo koji drugi uređaji koji podržavaju protokol za praćenje mreže (engl. *simple network management protcol*, skraćeno SNMP). Konfiguracija se odvija u dvije faze: prva je na *web-sučelju*, a druga na uređaju koji se promatra. Praćenje pojedinog poslužitelja počinje dodavanjem novog objekta (engl. *host*) kroz *web-sučelje* (kroz *configuration tab*), konfiguracijom načina praćenja te dodavanjem predloška (engl. *tameplate*) objektu. Objektu na *web-sučelju* moramo dodati ispravni IP i ime. Kada je odrađen dio na *web-sučelju*, onda je potrebno odraditi konfiguraciju na klijentu koji se promatra. Kako bismo omogućili poslužitelju prikupljanje podataka na *web-sučelju*, potrebno je postaviti identične vrijednosti kao i u konfiguracijskoj datoteci na klijentu kojeg pratimo.

Name ▲	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates
db1	Applications 11	Items 74	Triggers 23	Graphs 11	Discovery 2	Web	172.16.140.25: 10050	Template App Galera Cluster, Template OS Linux (Template App Zabbix Agent)

Slika 4.2 Izgled *hosta* na centralnom sučelju s obaveznim parametrima

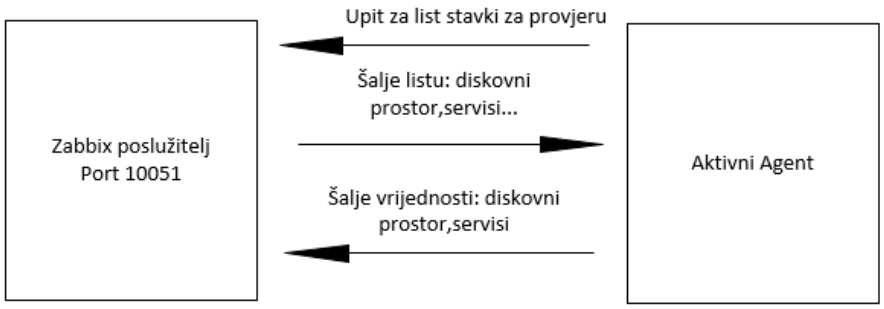
Druga faza počinje instalacijom *agenta* na poslužitelj. Najlakši je način za konfiguraciju i moguće je automatizirati proces instalacije na *Windows* i *Linux servere*. Aplikacija *Zabbix agent* ima veličinu od 700 KB i moguće ga je instalirati na više načina ručno ili korištenjem alata za udaljene instalacije (primjerice *System Center Configuration Manager* skraćeno SCCM[9] ili slični alati) ili *powershell* i *bash* skriptu ovisno o operacijskom sustavu na kojem se instalira.

[9] *SCCM* je alat za upravljanje informacijskim sustavom i kojim se može odrađivati grupna administracija računala.



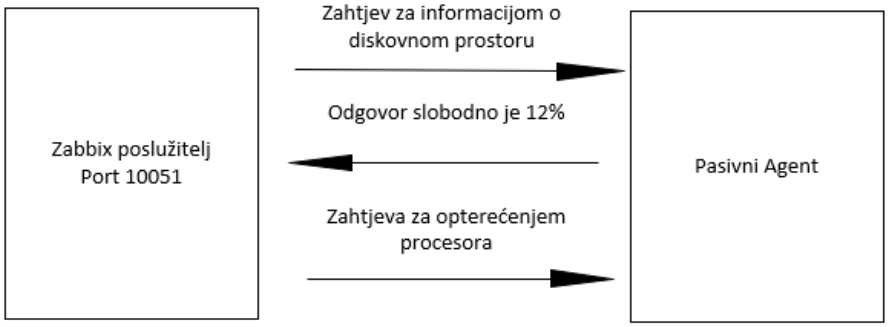
Slika 4.3Uspješno konfigurirano praćenje preko *agenta*

Postoje dva načina rada *agenta*. U aktivni način rada *agent* šalje upit poslužitelju za listu *itema* koje je potrebno slati. U njemu postoji *Zabbix poller* proces koji prikuplja podatke s udaljenih uređaja i poslužitelja. Također, *poller* služi za provjeru podataka koje je potrebno prikupljati. Povezivanje se ostvaruje sa *serverske* strane po *portu* 10051. *Zabbix trapper* proces kontrolira ostvarenu povezanost. *Agent* aktivno šalje vrijednosti *serveru* prema `RefreshActiveChecks` parametru (120 sekundi po zadanim postavkama) u konfiguracijskoj datoteci na objektu koji promatramo.



Slika 4.4 Princip rada aktivnog *agenta*

U pasivnom način rada *agent* ne šalju upite poslužitelju. Poslužitelj šalje upite *agentu* po potrebi. Prednost ovog način rada jest da je manje opterećenje na strani poslužitelja na kojem je instaliran *agent*.

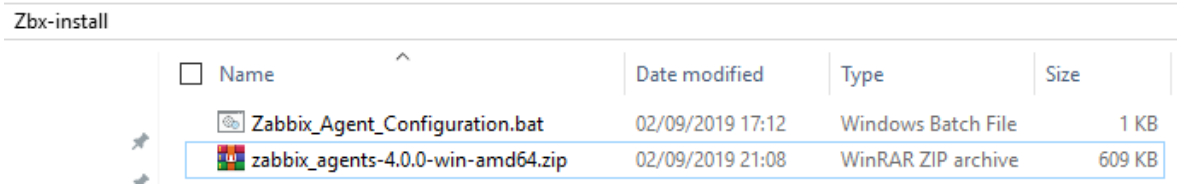


Slika 4.5 Princip rada pasivnog *agenta*

SNMP je protokol namijenjen za praćenje uređaja. Prikuplja informacije koje administratorima koriste za lakše identificiranje problema. Ovaj protokol koristimo kad nismo u mogućnosti instalirati *agent* primjerice ako se radi o uređaju s procesorskom arhitekturom na koji nije podržan *agent*, npr. mrežni preklopnih (engl. *switch*), baterija (engl. *Uninterruptible Power Supply*, skraćeno UPS), poslužitelj za skladištenje podataka (engl. *storage*) i sl. Na svakom objektu koji će se pratiti *Zabbix* serverom definira se SNMP sučelje te šalje jedinstvene vrijednosti identifikatore objekata (engl. *Object Identifier*, skraćeno OID) kojima prikuplja informacije. Njihove vrijednosti i sadržaji poslani su SNMP porukom. Međutim, takav način komunikacije nije moguće interpretirati ljudskim očima pa postoji centralna baza informacija (engl. *management information base*, skraćeno MIB) s koje će se prevesti informacije u formu iz koje se može izvući kontekst i vrijednost.

4.2.1. Podešavanje *agenta* na *Microsoft* poslužiteljima

Instalacija *agenta* jedna je od napornijih zadataka prilikom rada sa *Zabbix* poslužitelju. Moramo je na svakom od poslužitelja instalirati. Dobra je praksa svaki posao takve prirode automatizirati ako je to moguće. Na poslužitelju će se kreirati direktorij sa svime potrebnim za instalaciju. Koristeći *bash* i *powershell* skripte automatizirat će se procedura instalacije. *Powershell* skripta odradit će instalaciju *agent* aplikacije. *Bash* skripta koristit će se za uređivanje konfiguracijske datoteke koju je potrebno urediti tako da parametri ime lokalnog stroja (engl. *hostname*), IP *adresa Zabbix* poslužitelja za aktivni i pasivni način rada *agenta* budu ispravni. Glavna *powershell* skripta u sebi će pozivati izvršavanje *Batch* skripte. To je praktično ako budemo trebali mijenjati konfiguracijsku datoteku ponovno u budućnosti.



Slika 4.6 Sadržaj instalacijskog direktorija

U direktoriju naziva *Zbx-install* nalazi se *zip file* u kojem je instalacija *Zabbix agenta* za 64 bitna *Windows* računala. Zabbix_Agent_configuration.bat je *batch* skripta koja zapisuje vrijednosti u konfiguracijsku datoteku zabbix_agentd.win.conf.

```
@echo off
#zapisuje hostname racunala
Echo Hostname=%computername% >> "C:\Zabbix\zabbix_agentd.win.conf"
#zapisuje adresu servera za pasivni način rada
echo Server=172.16.140.103>> "C:\Zabbix\zabbix_agentd.win.conf"
#zapisuje adesu servera za aktivni nacin rada
Echo ServerActive=172.16.140.103>> "C:\Zabbix\zabbix_agentd.win.conf"
#mijenja lokaciju log file-a
echoLogFile=c:\zabbix\zabbix_agentd.log >>
"C:\Zabbix\zabbix_agentd.win.conf"
```

Kod 4.1 *Batch* skripta za uređivanje konfiguracije *agenta*

Potrebno je preuzeti zip datoteku sa službene *Zabbix* stranice na kojoj se nalaze *agenti* u svim formatima i verzijama. Glavna *powershell* skripta prvo odrađuje dekompresiju instalacijske *zip* datoteke te instalira *agenta* na željenu lokaciju. Nakon instalacije odradit će se potrebno propuštanje kroz lokalni vatrozid. Skripta je testirana na zadnja dva poslužiteljska operacijska sustava *Window Server 2016* i *Windows Server 2019*. Za starije verzije potrebna je prilagodba. Kako bismo mogli raspakirati *Zabbix Agent* arhivu dodaje se funkcija koja dekopresiranje *zabbix_agent-4.0.0-win-amd64.zip* datoteke (Slika 4.6 Sadržaj instalacijskog direktorija).

```
#dodavanje unzip funkcije
Add-Type -AssemblyNameSystem.IO.Compression.FileSystem
functionUnzip
{
    param([string]$zipfile, [string]$outpath)

    [System.IO.Compression.ZipFile]::ExtractToDirectory($zipfile,
$outpath)
}
```

Kod 4.2 *Powershell* funkcija za raspakiravanje arhive

Kako bi se osigurali u slučaju da servis postoji, napisana je provjera stanja servisa te pozivanje na brisanje servisa ako se nalazi na poslužitelju na kojem se instalira *agent.*

```
#ovaj dio nije nužan, ali je koristan ukoliko servis već postoji na
serveru
If ((Get-Service $serviceName).Status -eq 'Running' -or "Stopped") {
Set-Location "C:\Zabbix"
.\zabbix_agentd.exe --uninstall }
```

Kod 4.3 Provjera i brisanje ako servis već postoji

Kreirat ćemo direktorij na željenoj lokaciji i prebaciti datoteke `zabbix_agentd.exe` i `zabbix_agentd.win.conf` koje će se koristiti za instalaciju. Obrisat će se nepotrebne datoteke `bin` i `conf`.

```
#unzip
Unzip "C:\Zbx-INS\zabbix_agents-4.0.0-win-amd64.zip" "C:\Zabbix"

#kopiranje podataka na lokaciju instalacije
```

```
Copy-Item -Path "C:\Zabbix\bin\zabbix_agentd.exe" -Destination
"C:\Zabbix\"
Copy-Item -Path "C:\Zabbix\conf\zabbix_agentd.win.conf" -Destination
"C:\Zabbix\"
Remove-Item -Path "C:\Zabbix\bin"
Remove-Item -Path "C:\Zabbix\conf"
```

Kod 4.4 Prebacivanje potrebnih i brisanje nepotrebnih datoteka s lokacije *agenta*

Kako bi poslužitelj i *host* mogli uspostaviti komunikaciju potrebno je pokrenuti *batch* skriptu (Kod 4.1 *Batch* skripta za uređivanje konfiguracije *agenta*) koja će urediti konfiguracijsku datoteku.

```
#pozicioniranje na lokaciju
Set-Location "C:\Zbx-INS"

#pokretanje skrite za editiranje konfiguracijske datoteke
.\Zabbix_Agent_Configuration.bat
```

Kod 4.5 Pokretanje *batch* skripte za uređivanje konfiguracijske datoteke

Kada je konfiguracija datoteke odrađena, može se pokrenuti instalacija *agenta* koristeći `zabbix_agentd.exe` datoteku koja se nalazi na lokaciji `c:\zabbix`.

```
#postavi se na lokaciju za instalaciju
Set-Location C:\Zabbix

 #instlacija agenta
C:\Zabbix\zabbix_agentd.exe --config C:\zabbix\zabbix_agentd.win.conf --
install
sleep 2

#postavljanje servisa na automatsko pokretanje
Set-Service -Name "Zabbix Agent" -StartupType automatic
```

Kod 4.6 Instalacija agenta i postavljanje automatskog pokretanja servisa

Sve predradnje osigurane su osim propuštanja *portova* kroz vatrozid. Za komunikaciju između poslužitelja i *hosta* moramo propustiti sučelja 10050 i 10051 i pokrenuti servis inicijalno.

```
#vatrozid pravila
netsh advfirewall firewall add rule name="Open Zabbix agentd port 10050
inbound" dir=in action=allow protocol=TCP localport=10050
netsh advfirewall firewall add rule name="Open Zabbix trapper port 10051
inbound" dir=in action=allow protocol=TCP localport=10051

netsh advfirewall firewall add rule name="Open Zabbix agentd port 10050
outbound" dir=out action=allow protocol=TCP localport=10050
netsh advfirewall firewall add rule name="Open Zabbix trapper port 10051
outbound" dir=out action=allow protocol=TCP localport=10051
sleep 2

#pokretanje servisa
start-Service -Name "Zabbix Agent"
```

Kod 4.7 Propuštanje *agenta* kroz vatrozid i pokretanje servisa

4.2.2. Podešavanje *agenta* na *Linux* distribucijama

Istu skriptu moguće je napisati na *CentOS serveru*, a za pokretanje na drugim distribucijama potrebna je prilagodba. Skripta ima istu namjenu, ali je puno kraća i jednostavnija za pokretanje te nije potrebno preuzimati datoteku sa službenih stranice već to *yum* paketni sustav odradi sam. Grupnu konfiguraciju *agenta* lakše je odraditi na *Linux* distribucijama. Potrebno je dodati repozitorij kako bismo mogli preuzeti *agenta*.

```
#!/bin/bash
#dodavanje repozitorija za zabbix agenta
rpm -Uvhhttps://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-
4.0-2.el7.noarch.rpm

#skidanje paketa za zabbix agenta
Yum install zabbix-agent -y
sleep 2
```

Kod 4.8 Dodavanje repozitorija za *Zabbix agenta*

Isto kao i u prijašnjoj konfiguraciji istih parametara na *Microsoft* implementacijama potrebno je definirati i na *CentOS serveru*.

```
#zapisivanje imena poslužitelja u konfiguracijsku datoteku
echo Hostname=$HOSTNAME >> /etc/zabbix/zabbix_agentd.conf

#zapisivanje IP adrese poslužitelja za aktivan način rada
echo Server=172.16.140.100 >> /etc/zabbix/zabbix_agentd.conf

#zapisuje IP adresu poslužitelja za pasivan način rada
echo ServerActive=172.16.140.100 >> /etc/zabbix/zabbix_agentd.conf

#zapisuje lokaciju za log datoteku
echo LogFile=c:\zabbix\zabbix agentd.log >> "C:\Zabbix\zabbix agentd
sleep
```

Kod 4.9 Uređivanje konfiguracijske datoteke na *CentOS* poslužitelju

Također, potrebno je propustiti sučelja kroz vatrozid te omogućiti i pokrenuti servis.

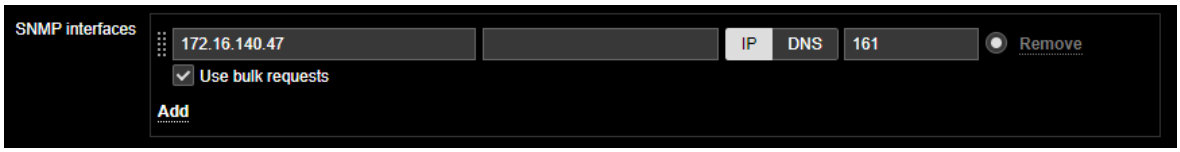
```
#konfiguracija pravila za lokalni vatrozid
firewall-cmd --add-service={http,https} --permanent
firewall-cmd --add-port={10051/tcp,10050/tcp} --permanent
firewall-cmd –reload

#pokretanje servisa i postavljanje automatskog pokretanja
Systemctl enable zabbix-agent
systemctl start zabbix-agent
```

Kod 4.10 Popuštanje sučelja kroz vatrozid

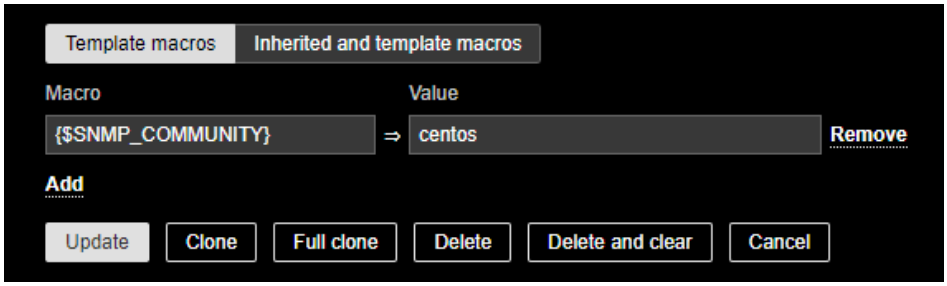
4.2.3. Podešavanje *SNMP* praćenja

Nakon kreiranja objekta na *web-sučelju* i podešavanja SNMP adrese potrebno je dodati predložak (engl. *tameplate*) preko kojeg će poslužitelj doći do vrijednosti koje mogu pronaći problem i poslati upozorenje.



Slika 4.7 Dodavanje SNMP sučelja na *hosta*

Već ugrađeni predlošci za navedenu vrstu praćenja mogu biti dovoljni uz prilagodbu okidača upozorenja. Također, za završavanje konfiguracije na *web-sučelju* potrebno je definirati riječ za autorizaciju (engl. *community string*). Ta vrijednost mora biti identična na oba objekta kako bi se mogle početi slati vrijednosti za praćenje pojedinih servisa. *Community string* može se definirati kao *macro* naredba na razini predloška ili objekta koji se prati. Ako je definiran na razini predloška, onda će se automatski primijeniti, a ako je na razini pojedinog objekta (pojedinog poslužitelja ili mrežnog uređaja), onda moramo za svaki objekt posebno postaviti vrijednost, u ovom slučaju je to šifra za autentifikaciju SNMP-a.



Slika 4.8 Definiranje *SNMP* šifre na *Zabbix* poslužitelju

Konfiguracija na klijentu koji ćemo pratiti demonstrirat će se na *Microsoft* virtualnom poslužitelju, ali princip konfiguracije je identičan i na drugim uređajima. Potrebno je omogućiti servis i dodati pripadajući *community string*. Kod *Microsoft* poslužitelja potrebno je definirati potonji na razini servisa.

```
#instalacija snmp srvisa na Windows poslužitelju
Install-WindowsFeature SNMP-Service -IncludeAllSubFeature -Verbose
```

Kada je servis dodan, automatski će se pokrenuti. Jedino što je sada potrebno napraviti jest dodavanje šifru u postavkama servisa koja mora biti ista kao i u predlošku.



Slika 4.9 Definiranje SNMP šifre na *Microsoft* poslužitelju

Ako je sve ispravno postavljeno, onda će nam se na centralnom sučelju pojaviti uspješan indikator SNMP praćenja isto kao i u slici niže (Slika 4.10 Uspješan indikator praćenja preko *SNMP-a*Slika 4.10 Uspješan indikator praćenja preko *SNMP-a*) *hosta.*



Slika 4.10 Uspješan indikator praćenja preko *SNMP-a*

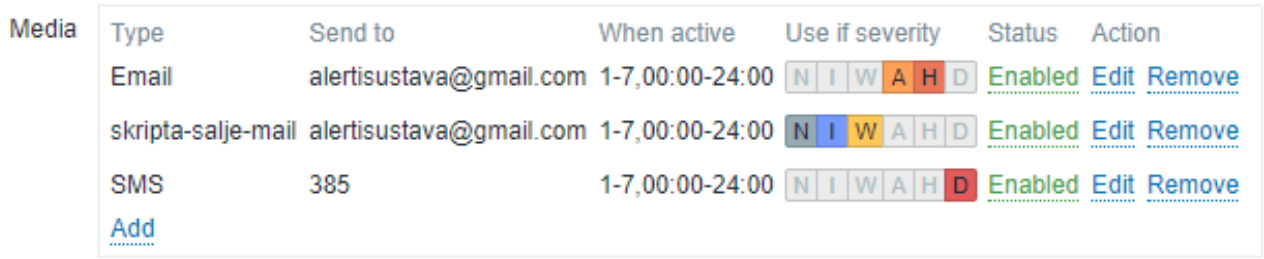
4.3. Udaljeno slanje obavijesti o problemima

Udaljeno slanje problema jest najvažnija funkcionalnost svakog servisa za praćenje sustava, nitko nema vremena neprekidno gledati u sučelje. Prilikom inicijalne konfiguracije sustava spomenuta funkcionalnost pomoći će nam u podešavanju okidača. Akcije uređivanja radit ćemo i odmah kada podignemo funkcionalni poslužitelj prema planu. Upozorenja različitim kanalima (*sms, e-mail*) u početcima životnog ciklusa sustava za praćenje koriste se kako bismo primijetili određena upozorenja koja dolaze učestalo. Kasnije će ova funkcionalnost biti od krucijalne vrijednosti u smislu obavještavanja ljudi koji mogu napraviti promjenu u trenutku kada se dogodi problem. Preduvjet je imati precizne okidače koji generiraju upozorenja (engl. *alert*), samo u slučaju kada se problem uistinu i pojavi. *Zabbix* poslužitelj upozorenja dijeli u nekoliko kategorija po riziku:

- *Notclassified* – postoji okidač, ali nije definirano koliki je utjecaj
- *Information* – koristi se za praćenje neke vrijednosti koja nije nužno upozorenje, ali je zanimljiva za promatranje, npr. učestalosti prijave korisnika
- *Warning* – moguć je utjecaj na sustav, najmanje je štetan za sustav

- *Average* – moguć je utjecaj na sustav, srednje je štetan za sustav
- *High* – moguć je utjecaj na sustav, visoko je štetan za sustav
- *Disaster* – dogodila se katastrofa.

Upozorenja se mogu slati koristeći elektroničku poštu, mobilnim porukama, preko *Jabbera*[10] ili skripte. Kanali se podešavaju kao tipovi medija (engl. *media types*) kroz administrativno *web-sučelje*. Potonji se povezuju s korisnicima kojima pridružuju mediji preko kojih će se slati upozorenja. Zadnji korak prilikom konfiguracije ove funkcionalnosti jest kreiranje akcije (engl. *action*), tj. definiranje koja će se upozorenja slati administratorima u slučaju pojave određenih upozorenja za objekte ili grupe objekata. Ovakav način slanja jest jezgrovit, moguće je filtrirati slanje prema ozbiljnosti (engl. *saverity*) upozorenja i prema grupi objekata. Primjerice, daje nam slobodu da se upozorenja za mrežne uređaje šalju samo dežurnom mrežnom administratoru, a upozorenja za virtualne i fizičke *servere* dežurnom administratoru operacijskih sustava. Također, možemo napraviti da se samo katastrofalni događaji šalju porukom. Jedini način da generira vrijednost od *Zabbix* sustave jest da se obavijesti šalju kada se stvarno problem i dogodi. Na taj način korisnik će biti zadovoljan, a administrator će moći reagirati samo kada je potrebna intervencija što znači više vremena za druge stvari.



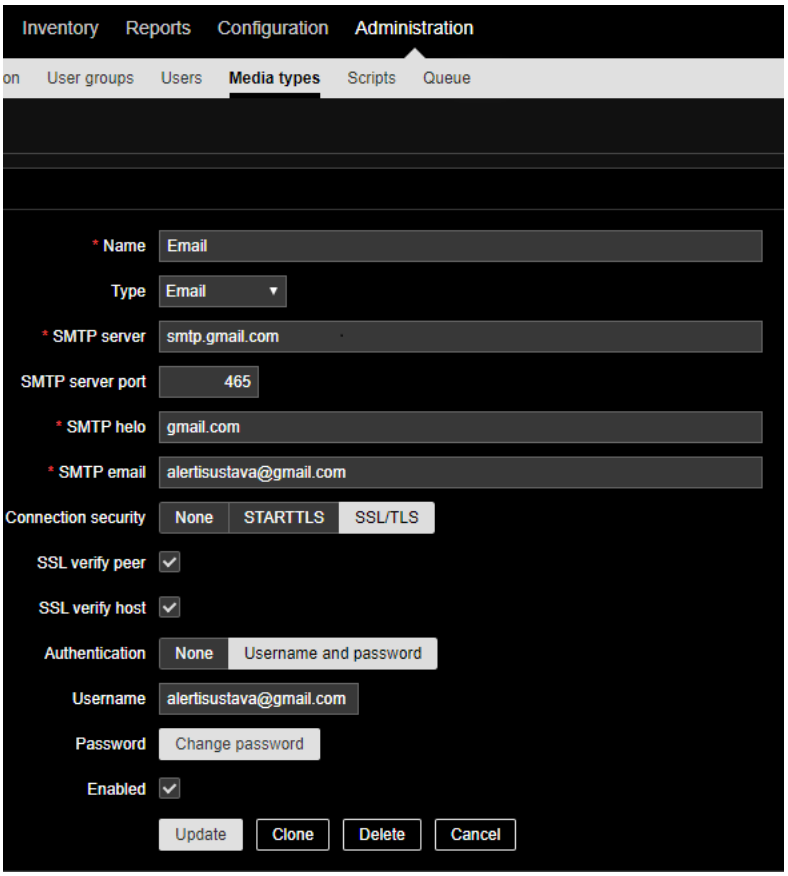
Slika 4.11 Konfiguracija medija za pojedinog korisnika

4.3.1. Podešavanje obavijesti pute elektroničke pošte

Navedeni način slanja obavijesti jest i najbitnija stavka i bez nje poslužitelj ne bi imao puno koristi u sustavu. *Zabbix* je moguće posložiti tako da šalje elektroničku poštu preko poslužitelja koji se nalazi na lokaciji poslovanja (engl. *on premise*). Nažalost, ovaj dio

[10]Jabber je tehnologija otvorenog koda za slanje poruka

ne možemo izvesti, nemamo poslužitelj za primanje i slanje e-pošte. Možemo se snaći na dva načina; javnim dostupnim servisom *gmail* ili koristeći skriptu koju ćemo postaviti na *Zabbix* poslužitelje. Konfiguracija prve opcije slanja jest jednostavna. Prvo moramo kreirati *gmail* račun koji će imati svrhu primanja obavijesti sa *Zabbix* poslužitelja te za račun omogućiti slanje poruka s nesigurnih aplikacija. To moramo napraviti jer *google* brani slanje s neautoriziranih poslužitelja zbog sigurnosti. Zatim na *web-sučelju* moramo urediti već postojeći *email media type* (ili kreirati novi) na *web-sučelju* na način kao na slici niže te uključiti akciju `Report problems to Zabbix administrators.`



Slika 4.12 Konfiguracija primanja upozorenja *gmailom*

Isto je moguće postići na malo kompleksniji način preko skripte koja će se nalaziti lokalno na zbx1 i zbx2. Potrebno je skinuti i instalirati servise za slanje elektroničke pošte te urediti konfiguracijsku datoteku.

```
yum install ssmtp mailx -y
#uređivanje konfiguracijske datoteke za sstp
nano /etc/ssmtp/ssmtp.conf
#bitne stavke koje je potrebno urediti
root=alertisustava@gmail.com
```

```
mailhub=smtp.gmail.com:587
rewriteDomain=zavrnsni.local
hostname=zbx1 #promijeniti na zbx2 serveru
UseTLS=Yes
UseSTARTTLS=Yes
AuthUser=alertisustava@gmail.com
AuthPass=sifragmail
FromLineOverride=YES
```

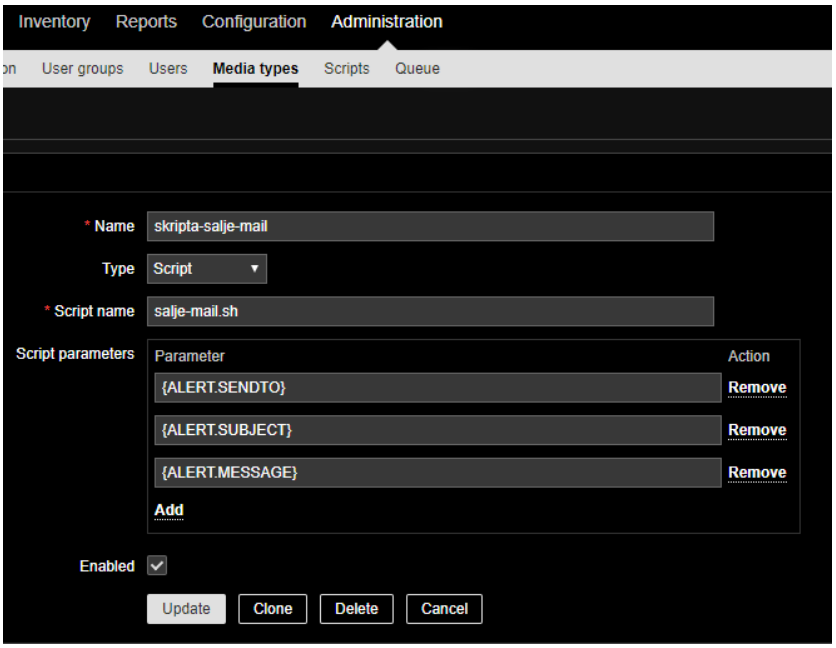
Kod 4.11 Konfiguracija slanja *mail* obavijesti preko skripte

U direktoriju na lokaciji `/usr/lib/zabbix/alertscripts` kreirat ćemo skriptu za slanje *maila*. Nazvat ćemo je *salje-mail.sh* i dodat ćemo prava da se može izvršiti

```
chmod +x salje-mail.sh.
#!/bin/bash
to=$1
subject=$2
body=$3
cat<<EOF | mail -s "$subject" "$to"
$body
EOF
```

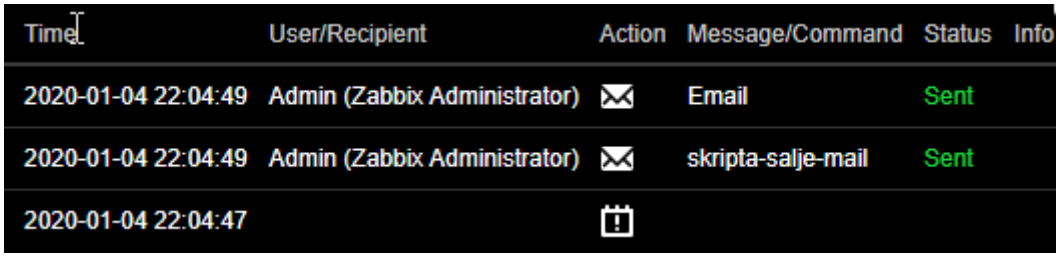
Kod 4.12 Skripta za slanje *mail* obavijesti

Na *web-sučelju* kreirat će se novi tip medija (engl. *media types*) preko kojeg će poslužitelj slati obavijesti skriptom. Ime skripte mora biti isto kao i na poslužitelju. Slanje neće funkcionirati ako se ne definiraju parametri koji su zapravo varijable u skripti na poslužitelju. Također, u administraciji korisnika nužno je da korisnik koji će primati e-poštu ima definiran medij (u ovom slučaju adresu alertisustava@gmail.com) te postavku u koje vrijeme će se slati obavijesti za pojedinu definiciju problema.



Slika 4.13 Konfiguracija poslužitelja za slanje *mail* obavijesti preko skripte

Ako su tipovi medija, akcije i korisnički medij postavljeni kako treba, onda za svaki problem moramo dobiti obavijest e-poštom te se zapis poslano (engl. *sent*) treba vidjeti u desnom kutu svakog upozorenja.



Slika 4.14 Provjera uspješnosti slanja

4.3.2. Podešavanje obavijesti putem *SMS-a*

Za katastrofične događaje moramo osigurati da administrator vidi poruku u svakom trenutku. Ljudska tendencija jest da se ne čita poslovna e-pošta iza radnog vremena pa ćemo podesiti slanje SMS porukom u slučaju da se dogodi događaj katastrofične ozbiljnosti (engl. *disaster alert*). Prilikom navedene implementacije postoji više načina za postizanje funkcionalnosti slanja mobilnih poruka obavijesti. Moguće je kupiti *Global System for Mobile Communications* (skraćeno, GSM[11]) modem te ga prezentirati virtualnoj mašini spajanjem na fizički poslužitelj preko serijskog sučelja. Prema dokumentaciji *Zabbix* testiran

[11]Standard za mobilne mreže.

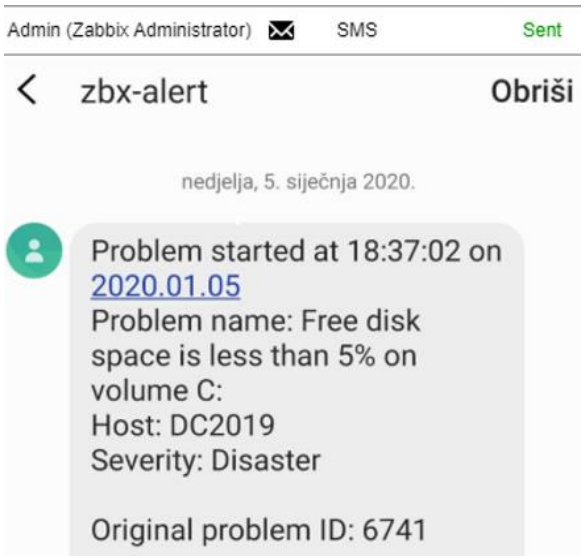
je sa *Siemens MC35* i *Teltonika Modem COM/G10.* No, postoje i implementacije kada nismo u mogućnosti doći do fizičke mašine, npr. što ako je *Zabbix* podignut u nekom od usluga računarstva u oblaku (engl. *cloud*) npr. *Azure-u*. U tome slučaju konfigurirali bismo slanje *SMS-a* koristeći skriptu koja se spaja na aplikacijsko programsko sučelje (engl. *application programming interface*, skraćeno API) servisa za slanje SMS poruka. Obavit ćemo registraciju na D7SMS[12] *web-stranici*. U direktoriju na lokaciji `/usr/lib/zabbix/alertscripts` ćemo kreirat ćemo skriptu za slanje *maila* preko D7SMS javnog servisa.

```
#pozicioniranje na lokaciju gdje se nalaze skripte za alerte
cd /usr/lib/zabbix/alertscripts
#preuzimanje skripte s interneta
wgethttps://raw.githubusercontent.com/d7networks/zabbix/maste
r/d7sms.py
#dodavanje prava izvršavanja skripte
chmod +x /usr/lib/zabbix/alertscripts/d7sms.py
#uređivanje skripte
vim /usr/lib/zabbix/alertscripts/d7sms.py
# moramo ažurirati ove linije u skripti sa podacima našeg
profila
    USERNAME = 'API_Username'# korisničko ime za API
    PASSWORD =  'API_Password'# zaporka za API
SOURCE_ADDRESS = 'zbx-alert' # ime pošiljatelja
```

Kod 4.13 Konfiguracija slanja SMS porukama

Također, u administraciji korisnika nužno je da korisnik koji će primati SMS ima definiran broj mobilnog uređaja (mora biti s brojem države u slučaju Hrvatske 385) te postavljeno u koje vrijeme će se slati obavijesti za probleme i katastrofalnu (engl. *disaster*) ozbiljnost. Ako su tipovi medija, akcije i korisnički medij postavljeni kako treba, onda za svaki problem katastrofalne ozbiljnosti moramo dobiti obavijest mobilnom porukom te zapis poslano (engl. *sent*) treba biti vidljiv u desnom kutu svakog upozorenja.

[12]Javni servis za slanje *SMS* poruka.



Slika 4.15 Potvrda uspješnog slanja i prikaz izgleda poruke

4.4. Podešavanje predložaka, grafičko prikazivanje i *proxy* implementacija

U ovom poglavlju pojasnit će se pojmovi koji su potrebni za uređivanje predložaka i pokazati uređivanje predložaka na konkretnom primjeru. Prema spomenutoj logici može se pristupiti uređivanju vrijednosti bilo kojeg predloška.

4.4.1. Uređivanje predložaka

S trenutnom konfiguracijom sustava najvjerojatnije imamo puno e-pošte o događanjima na sustavu koji promatramo. Sad slijedi faza podešavanja svakog predloška kako bi nam se javljale informacije koje su uistinu potrebne. Predložak (engl. *tamplate*) je skupina različitih entiteta koje primjenjujemo na objekte koje promatramo. Ti entiteti su:

- stavka (engl. *item*) prikuplja podatke s objekata koji se prate. Način da se doda više stavki odjednom moguće je apliciranje predloška.
- Okidač (engl. *trigger*) logičke vrijednost koja procjenjuje prikupljene podatke i reagira ako je definirani prag (engl. *threshold*).
- Graf (engl. *graphs*) s mnoštvom podataka *Zabbix* ima mogućnost vizualne prezentacije različitim grafovima.

- Aplikacija (engl. *application*) je skupina različitih stavki po kategorijama; primjerice predložak *OS Windows* ima aplikaciju naziva „CPU“ koja se sastoji od stavki za praćenje procesora.
- Ekrani (engl. *screen*) tablica je namijenjena za prezentiranje informacija.
- Pravilo otkrivanja (engl. *discovery rule*) koristi se za otkrivanje objekata i servisa.

Predloške je moguće preuzimati, kopirati, povezivati i uređivati. Svaki od predložaka ima svoje okidače koji imaju svoje predefinirane vrijednosti. Svaki okidač se isto tako može postaviti na željenu vrijednost. Proceduru uređivanja predložaka pokazat će se na postojećem predlošku koji je dostupan odmah iza instalacije i služi za praćenje poslužitelja koji imaju *Windows* server operacijski sustav. *Template* naziva OS *Windows* ima sve potrebno za praćenje *serverskih* komponenti, ali moramo posložiti vrijednosti okidača kako bismo isti predložak mogli koristiti. Primjerice, kada bismo postavili isti predložak na grupu objekata (engl. *hostgroup*), *Zabbix* će nam javiti ako postotak slobodnog prostora na disku prijeđe definirani okidač, poslat će se upozorenje srednje ozbiljnosti kada ostane 20 posto slobodnog prostora. Navedena vrijednost nije prihvatljiva zato što će upozorenje biti generirano prerano, a kasnije kad situacija postane ozbiljna nećemo imati sve potrebne informacija o stanju diskovnog prostora. Preko spomenutog predloška *Zabbix* otkrije diskove preko pravila otkrivanja naziva *Mounted file system discovery* koje koriste regularni izraz[13](engl. *regular expresion*).

```
^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$[Result is TRUE]
```

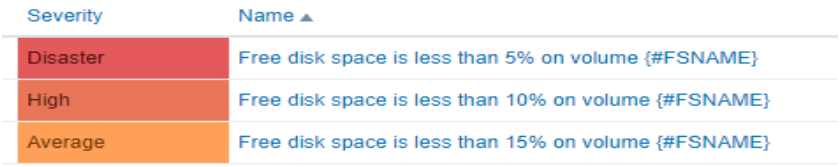
Kada se identificiraju diskovni sustavi, onda se primjenjuje prototip okidača (engl. *trigger prototypes*) koji ima sljedeći izraz. Prvi dio izraza jest stavka (engl. *item*), a drugi je okidač koji je definiran na dvadeset posto.

```
{Template OS Windows:vfs.fs.size[{#FSNAME},pfree].last(0)}<20
```

Stavka provjerava dostupni prostor na disku i okidač posljedično šalje upozorenje sukladno definiciji (manje od dvadeset posto). Nakon što smo shvatili način na koji predložak funkcionira, možemo nadodati svoje okidače koji će odgovarati našim potrebama. Poželjno je da se srednje upozorenje generira na petnaest, visoko na deset, a katastrofalno

[13]Skup znakova definiran sintaksom koji služi za pretraživanje.

na pet posto slobodnog prostora na disku. Tako će administratori sustava biti navrijeme obaviješteni o stanju, slat će se više obavijesti i imat ćemo vremena za reagiranje.



Slika 4.16 Primjer definiranja *triggera*

Za provjeru servisa također se koristi pravilo otkrivanja (engl. *dicovery rule*), po regularnom izrazu `^(automatic|automaticdelayed)$[Resultis TRUE]` provjeravaju se svi servisi koji imaju status automatskog pokretanja i pokretanja s odgodom. Pravilo koristi stavku koja provjerava trenutno stanje servisa i upozorenje se dogodi ako servis nije upaljen više od tri minute.

```
{Template OS Windows:service.info[{#SERVICE.NAME},state].min(#3)}<>0
```

Uz navedeno upozorenje dodat ćemo i dodatna dva: ako se dogodi da servis bude nedostupan više od osam minuta generirat će se upozorenje visoke ozbiljnost, a za isti događaj trajanja dvadeset minuta definirat će se upozorenje katastrofalnih razmjera. Također, ako iz nekog razlog ne želimo da se prati određeni servis, možemo ga izbaciti definiranjem regularnog izraza za pravilo otkrivanja za servise na *Microsoft* platformi (engl. *windows server names for discovery*). Izraz mora imati netočan (engl. *false)* rezultat što znači da će ga pravilo preskočiti u otkrivanju i *Zabbix* neće definirati upozorenja za takve servise. Primjerice, definira se izraz koji ima vrijednost `^(SCardSvr)$[Resultis FALSE]` i znači da se servis `SCardSvr` neće pratiti. Ako se dogodi da pojedini okidač javlja netočne informacije i da prikupljanje takve informacije nema koristi, možemo u potpunosti onemogućiti takve okidače i isti se neće više pojavljivati.

Odrađivanja uređivanja okidača definirati može i prilikom pojave nekog problema npr. učestalo se pojavljuje neki problem koji nije točan. Odabirom problema može se provjeriti konfiguracija te urediti prije spomenute postavke za svaku obavijest o problemu posebno. Poželjno je da na infrastrukturi postoji jedna virtualna mašina čija će uloga biti isključivo za testiranje. Ako postoji, onda možemo na njoj inducirati upozorenja različitim naredbama i provjeriti zadovoljava li praćenje definirana očekivanja (Kod 4.14 Komande za induciranje upozorenja.

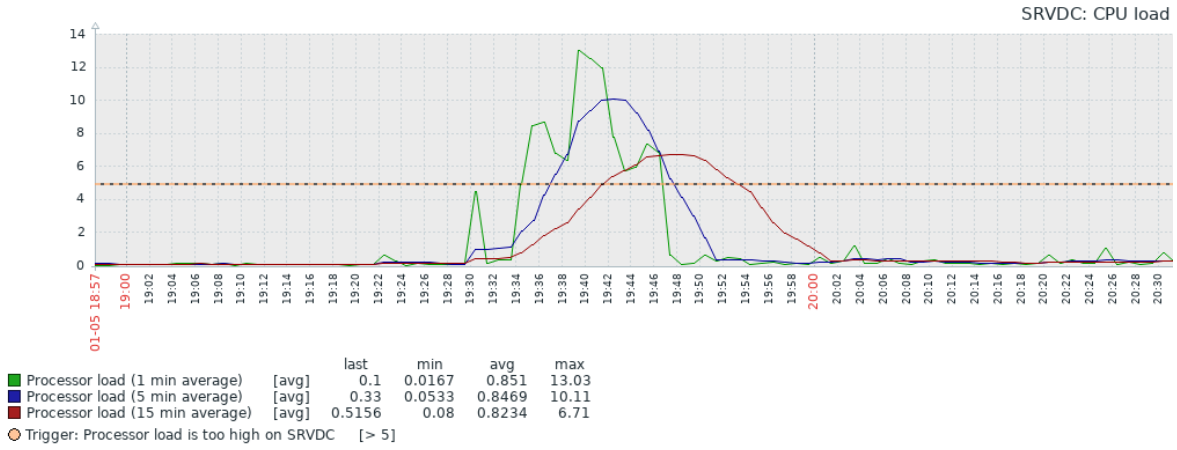
```
#napravi petlju kojom se CPU opteretiti
do{$p += get-process} until (!$p)
```

```
#kreira tekstualni podataka veličine 90GB i zapuni disk
fsutil.exe file createnewveliki_podatak.txt -path C:\ 90000000000
```

Kod 4.14 Komande za induciranje upozorenja

4.4.2. Grafičko prikazivanje podataka

Osim što pruža usluge praćenja sustava *Zabbix*, poslužitelj sve vrijednosti stavki koje prikuplja sprema u bazu podataka. Vrijednosti iz baze mogu se prikazati grafički. Primjerice, ako nas zanima koliko je bilo opterećenje procesora za neki poslužitelj koji je u nadzoru, moguće je pregledati *web-sučelje* i moramo odabrati graf (engl. *graphs*). Za objekt koji promatramo odaberemo vrijednost koju želimo promatrati i vremenski interval za koji želimo vidjeti grafikon. Vremenski interval praćenja moguće je definirati do sekunde točnosti i vrijednosti su opisane u legendi. Također, možemo vidjeti kada će se generirati upozorenje točnije na koliko je definiran okidač. Na slici niže (Slika 4.17 Primjer grafičkog prikazivanja) vidi se da je od 19:41 do 19:54 prosjek opterećenja procesora unutar petnaest minuta porastao preko vrijednosti definirane okidačem.



Slika 4.17 Primjer grafičkog prikazivanja

Ako je potrebno prikazati više različitih grafova na jednom centralnom mjestu, može se složiti preko ekrana (engl. *screens*). Korisno je neke informacije pratiti na centralnom sučelju kako bi se upoznalo ponašanje sustava duže vrijeme (engl. *baseline*), ali može biti i od vrijednosti za korisnika ako ga zanima kretanje neke specifične vrijednosti, npr. grafički prikaz broja poslanih elektroničke pošte unutar mjesec dana za pojedini poslužitelj.

Također, moguće je instalirati dodatak na naše poslužitelje koji će omogući dodatne funkcionalnosti analize i prezentiranja podataka. Na zbx1 i zbx2 poslužitelje instalirat ćemo

Grafana dodatak. Navedeno rješenje otvorenog koda za grafičko prezentiranje spojit ćemo na *Zabbix* servis kako bismo omogućili detaljnije prezentiranje različitih vrijednosti. *Zabbix* servis prikupljat će podatke, *Grafana* servis iste će te podatke prezentirati na centralno sučelje automatski.

Kako bi se mogli skinuti paketi potrebno je dodati repozitorij.

```
#dodavanje repozitorija
vim /etc/yum.repos.d/grafana.repo
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

Kod 4.15 Dodavanje repozitorija za *Grafanu*

Moraju se skinuti sve potrebni paketi te pokrenuti i omogućiti servisi. Također, moramo propustiti TCP sučelje 3000 koje koristi *Grafana.*

```
yum install grafana fontconfig freetype* urw-fonts -y
#pokretanje servisa
systemctl start grafana-server
systemctl enable grafana-server.service
firewall-cmd --zone=public --add-port=3000/tcp --permanent
firewall-cmd-reload
```

Kod 4.16 Skidanje paketa i pokretanje servisa

Kada je inicijalna konfiguracija Grafane odrađena, možemo instalirati pakete za spajanje na *Grafane* na *Zabbix*.

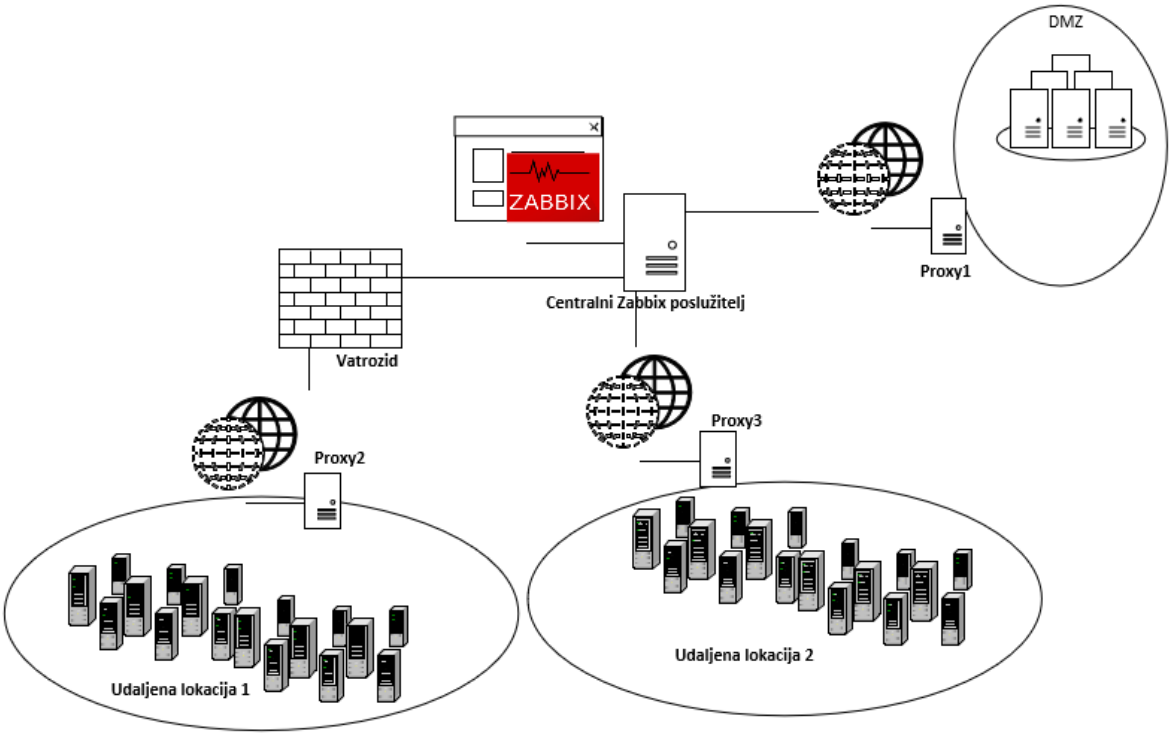
```
#instalacija plugina za Zabbix
grafana-cli plugins install alexanderzobnin-zabbix-app
#ponovno pokretanje servisa nakon insalacije
systemctl restart grafana-server
```

Kod 4.17 Integracija sa *Grafane* sa *Zabbixom*

4.4.3. *Zabbix Proxy* implementacija

Zabbix proxy služi za skupljanje podataka od objekata i njihovo slanje na centrali poslužitelj, on radi u ime poslužitelja kako bi smanjio opterećenje (engl. *load*). Spomenuta komponenta nije obavezna za implementaciju, ali može biti od koristi ako se prati udaljene lokacije koja ima puno objekata koje je potrebno motriti ili imamo lošu povezanost prema lokaciji pa je funkcionalnost potrebna. Također, na ovaj način poslovanja bi pratila više odvojenih sustava u smislu održavanja više različitih korisnika preko jednog centralnog poslužitelja. Kod svakog korisnika podigao bi se poseban *proxy*.

Postoji dva načina rada, slično kao i kod *agenta*, aktivni i pasivni način. Kod aktivnog načina *proxy* će se spajati na poslužitelj i tražiti konfiguracijske podatke. Kod pasivnog će se poslužitelj spajati na *proxy*.



Slika 4.18 *Zabbix Proxy* implementacija

Konfiguracija se odvija na sličan način kao kad bi se podizao pojedini *Zabbix* poslužitelj. Bitno je preuzeti ispravnu verziju paketa, kreirati bazu podatka, obaviti podešavanje konfiguracijske datoteke u kojoj su bitni parametri ime *proxy* poslužitelja, ime baze i vjerodajnice korisnika koji ima prava na bazu te IP centralnog *Zabbix* poslužitelja. Servis moramo propustiti kroz vatrozid i antivirus. Zadnja akcija jest dodavanje objekta na

web-sučelju. Konfiguracija započinje dodavanje repozitorija na poslužitelj te skidanjem potrebnih paketa za izvedbu *Zabbix Proxy servera*.

```
rpm --import http://repo.zabbix.com/RPM-GPG-KEY-ZABBIX
rpm -ivhhttp://repo.zabbix.com/zabbix/4.0/rhel/7/x86_64/zabbix-release-4.0-2.el7.noarch.rpm
#instalacija i preuzimanje proxy paketa
Yum install zabbix-proxy-mysql -y
```

Kod 4.18 Dodavanje repozitorija i skidanje paketa za *proxy*

Također, kako će *proxy* imati svoju lokalnu bazu podataka moramo je skinuti i instalirati na poslužitelj. Također, na bazu ćemo dodati prava korisniku te dodati inicijalnu shemu za *proxy*.

```
#instalacija i preuzimanje paketa za bazu podataka
Yum install mariadb-server -y
#instalacija MariaDB
mysql_secure_instalation
#ulazak u mysql komandnu liniju
mysql -u root -p
#kreiranje praze baze podataka
MariaDB [(none)]>create database proxy_baza character set utf8 collate utf8_bin;
#dodavanja prava na bazu podataka
MariaDB [(none)]>grant all privileges on proxy_baza.* to zabbix@localhost identified by "centos";
#primjena prava
MariaDB [(none)]>flush privileges;
#ubacivanje inicijalne šeme u bazu
zcat /usr/share/doc/zabbix-proxy-mysql-4.0.2/schema.sql.gz | mysql -u zabbixuser zabbix_proxy -p
```

Kod 4.19 Skidanje, instalacija i konfiguracija *proxy* baze podataka

Predradnje instalacije servisa i baze su gotove. Sada možemo urediti *proxy* konfiguracijsku datoteku u kojem ćemo navesti IP *adresu* centralnog *Zabbix* poslužitelja te vjerodajnice za spajanje na lokalnu bazu. Pokrenut ćemo i omogućiti servis.

```
#uređivanje konfiguracijske datoteke
vi /etc/zabbix/zabbix_proxy.conf
Server=172.16.140.103
Hostname=zabbix-proxy
DBName=proxy_baza
DBUser=zabbix
```

```
DBPassword=centos
#inicijalno porketanje servisa
Systemctl enable zabbix-proxy
systemctl start zabbix-proxy
```

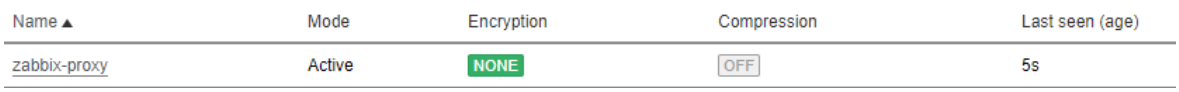
Kod 4.20 Konfiguracija *proxy* servisa

Za kraj konfiguracije poslužitelja potrebno napraviti propuštanja kroz sigurnosni modul i vatrozid.

```
#propuštanje kroz SELinux
cat /var/log/audit/audit.log | grepproxy | grepdenied | audit2all
semodule -i zabbix_proxy.pp
setsebool -P zabbix_can_network=1
#konfiguracija firewall pravila
firewall-cmd --permanent --add-port=10050/tcp
firewall-cmd --permanent --add-port=10051/tcp
```

Kod 4.21 Popuštanje kroz *SELinux* i vatrozid

Kada je konfiguracija poslužitelja obavljena, možemo preći na *web-sučelje* i dodati novi *proxy* objekt. U trenutku kada je *proxy* vidljiv, a to se može provjeriti po *last seen* vrijednosti u sekundama (Slika 4.19 Uspješna konfiguracija *Zabbix proxy* poslužitelja) onda možemo krenuti u konfiguraciju *hostova* preko istog *proxy* poslužitelja.



Slika 4.19 Uspješna konfiguracija *Zabbix proxy* poslužitelja

5. Usporedba sa aplikacijama za *logiranje*

U ovom poglavlju usporedit će funkcionalnosti *Zabbix* poslužitelja sa sustavima koji su već dostupni i vode zapisivanje događanja po poslužiteljima ili na udaljenom poslužitelju. Svaki poslužitelj i servis mora zapisivati krucijalne događaje vezane za funkcioniranje operacijskog sustava i aplikacija koje su na njemu instalirane. Kada će se doticati *Microsoft* tehnologija, onda će se pričati o *Event Vieweru*, a kada će se doticati *Linux* distribucija, pričat će se o *Syslogu*.

5.1. Usporedba s *Microsoft Event Viewer* sustavom

Windows Event Viewer prikazuje poruke o događanjima na strojevima koji imaju instaliran *Microsoft* operacijski sustav. Prva je aplikacija koja se pali ako je potrebno analizirati uzroke problema koji se dogodio na klijentskom računalu ili na nekom od poslužitelja. Prate se poruke o aktivnosti po ozbiljnosti događaja postoje greške (engl. *error*), upozorenja (engl. *warning*), informacije (engl. *information*). Događaji su grupirani u više kategorija, a svaka većina instalirani aplikacija dolazi sa svojom posebnom kategorijom. Svaki od događaja na sustavu ima svoj broj za raspoznavanje (engl. *identifire*) preko kojeg se ono može detaljnije istražiti. Moguće je postaviti spremanje više događaja od zadanih dvadeset *megabytea* koje *Event Viewer* sprema. Također, mogu se definirati i politike arhiviranja.

Međutim, svaki *Event Viewer* dolazi predefinirano s puno lažnih (engl. *false positiv*) uzbuna tako da ovaj alat nije dovoljno stabilan da bi se koristio kao sustav za praćenje. Također, sva događanja vidljiva su na stroju lokalno. Moguće je preko skripte slati poruke o događanjima na sustavu, ali za to je potrebno puno više vremena od instalacije *Zabbix agenta* na poslužitelj koji se promatra. Preveliko je administrativno vremensko opterećenje za konfiguraciju navedenog načina praćenja jer se koristi samo kada nema drukčiji izbor osim slanja određenog identifikatora (skraćeno, ID-a) nekog događaja e-poštom, a takvi su problemi rijetki s obzirom na količinu različitih rješenja za praćenje sustava. *Zabbix* nam za razliku od *Event Viewera* nudi centralno sučelje gdje se mogu pregledavati više odvojenih uređaja različite funkcije na jednom mjestu. Ima veću retenciju s obzirom na to da se sve promjene čuvaju u bazi podataka te ima mogućnosti da se iste te vrijednosti prezentiraju grafički. Za svako upozorenje koje *Zabbix* otkrije moguće je poslati obavijest različitim

kanalima što kod *Event Viewera* nije slučaj. *Event Viewer* koristio bi se nakon što nas *Zabbix* obavijesti da postoji problem na određenom poslužitelju i poslužio bi nam kao alat za detaljnu analizu. Primjerice, dogodi se zastoj nekog servisa. Administrator će se spojiti na server i napraviti reaktivne akcije kako bi se problem riješio, a poslije će se istraživati po *Event Vieweru* kako bi se vidjelo zašto se dogodio zastoj servisa i koje je trajno rješenje da se isti problem ne ponovi.

5.2. Usporedba s *Linux syslog* servisom

Syslog servis prima i procesira *Syslog* poruke na *Linux* distribucijama. *Log* poruke se na lokalnom *serveru* nalaze na lokaciji /var/log. Primjerice, status *zabbix-server* servisa možemo pregledati naredbom `tail -l /var/log/zabbix/zabbix_server.log`. U datoteci `zabbix_server.log` je zapisano trenutno stanje servisa. Ispis sadrži sve potrebne informacije koje mogu koristiti administratoru, ali problem je što zahtjeva ručnu provjeru i spajanje na uređaj.

Međutim *Syslog* može i imati kontekst *System Logging* protokola koji se koristi za praćenje sustavnih poruka na različitim uređajima. Njegova primarna svrha jest skupljanje zapisa s različitih uređaja na centralni poslužitelj kako bi se izbjegla potreba za spajanjem na svakom serveru posebno u svrhu pregleda *logova*. Koristi *user datagram protocol* (skraćeno, UDP) protokol na sučelju 514, ali često se koristi i *transmission control protocol (*skraćeno, TCP) 1468. Konfiguracija centralnog poslužitelja i preusmjeravanja *logova* ne predstavlja problem, ali i dalje ne odgovara funkcionalnostima koje ima *Zabbix* poslužitelj. Također, zapise koji su preusmjereni na centrali, *server* je potrebno podesiti tako da se prikazuju samo oni koji su potrebni, što zahtijeva dosta angažmana administratora. Korisno je ovaj protokol koristiti kao nadopunu sa *Zabbix* poslužiteljem. Primjerice ako *Zabbix* generira upozorenje, onda treba provjeriti centralni *syslog* server za provjeru *log* zapisa. *Syslog* poslužitelj jest odvojena funkcionalnost i ima svoje prednosti, ali ne može nas obavijestiti precizno različitim kanalima ako stvarno postoji problem na sustavu. Koristit ćemo ga kao komplementaran alat ako za to bude potrebe s obzirom na to da nam *Zabbix* poslužitelj šalje obavijesti o problemima.

Zaključak

Zabbix kao *server* solidan je alat i nudi sve potrebne funkcionalnosti kako bi se pratile komponente sustava. Velika prednost opisanog rješenja jest što ne zahtijeva licenciranje. Logička podjela objekata koji se prate kroz sustav jest skalabilna i daje slobodu za podešavanje na način koji je specifičan za pojedini sustav. Administrator ne mora neprekidno provjeravati sustav jer poslužiteljski servis to radi za njega te po potrebi se može spojiti i pogledati centralno sučelje. *Zabbix* i bazu podataka na koju se povezuje moguće je konfigurirati na jednom ili više virtualnih poslužitelja. Visoka dostupnost na razini virtualnih mašina moguće je konfigurirati koristeći dostupne besplatne servise na distribuciji *CentOS* i drugim *Linux* operacijskim sustavima. S obzirom na to da se svi podaci spremaju u jednu bazu podatka, možemo imati više aplikativnih poslužitelja, a bazu možemo smjestiti u *cluster* poslužitelja. Dizajn visoke dostupnosti, opisane u trećem poglavlju rada, uz prilagodbu može se implementirati za bilo koju aplikaciju na *Linux* distribucijama. Praćenje komponenti može se podesiti za bilo koji objekt za koji postoji predložak ili mogućnost praćenja putem SNMP-a. Dodavanje objekata u nadzor poglavito *hostova* koji nisu distribucije *Linuxa* (lako se skriptira) jest jedan od nedostataka *Zabbixa* zato što zahtijeva dosta vremena s obzirom na to da se mora obaviti u dva navrata; prvo na *web-sučelju*, a zatim na samom *hostu* koji se promatra, osim ako ne koristimo neki alat za njegovu automatizaciju što zahtijeva dodatne troškove. Bilo bi puno lakše da se konfiguracija *agenta* može obaviti samo jednim korakom, da se ono udaljeno instalira prilikom dodavanja *hosta* na *web-sučelju*. *Zabbix* možemo konfigurirati da nam udaljeno javlja o problemima na sustavu preko elektroničke pošte ili mobilne poruke što je najbolja funkcionalnost navedenog sustava.

Cilj je reagirati na problem kad se dogodi, a na taj način administrator će moći doći do obavijesti kad se dogodi. Obavijesti e-poštom mogu se namjestiti besplatno, a za SMS će se trebati uložiti u GSM modem ili zakupiti uslugu slanja poruka. Također, jedna od glavnih prednosti *Zabbixa* jest da se svi prikupljeni podaci mogu prezentirati grafički, a to se može iskoristiti za različite prezentacije i izvlačenja konteksta. Ako se radi o velikoj infrastrukturi na više lokacija, *proxy* poslužitelj bit će od koristi zbog smanjenja opterećenja na centralni server. Za implementaciju ovakvog sustava potrebno je imati inženjere koji će znati podesiti nesmetani rad sustava bez pogrešnih upozorenja. Sučelje na početku korištenja nije intuitivno te treba dosta vremena kako bismo se s njim upoznali. Postoji dostupna

dokumentacija na internetu i razvijena zajednica korisnika navedenog proizvoda pa je moguće doći do rješenja ako se dovoljno istraži.

Popis kratica

VIP	*Virtual IP*	virtualna *IP adresa*
VM	*Virtual Machine*	virtualna mašina
SE	*Security-Enhanced Linux*	antivirusni zaštita na *CentOS*u
NVPS	*new values per second*	nove vrijednosti u sekundi
SNMP	*Simple Network Management Protocol*	protokol za praćenje uređaja
AWS	*Amazon Web Services*	naziv Amazonove *colud* usluge
CIB	*Cluster Information Base*	glavna baza informacija u *clusteru*
SCCM	*System Center Configuration Manager*	Microsoftov alat za praćenje sustava
UPS	*Uninterruptible Power Supply*	baterija za pričuvu
OS	*Operating System*	operacijski sustav
GSM	*Global System for Mobile Communications*	standard za mobilnu komunikaciju
TCP	*Transmission Control Protocol*	protokol za prijenos podataka
UDP	*User Datagram SProtocol*	protokol za prijenos podataka

Popis slika

Slika 3.1 Izvedba visoke dostupnosti 7

Slika 3.2 Arhitektura *pacemakera[12]* 9

Slika 3.3 Očekivani ispis nakon pokretanja *clustera* 11

Slika 3.4 Pokretanje servisa nakon uspješnog *setupa* 12

Slika 3.5 Funkcionalno stanje *pc sclustera* s podešenim resursima 14

Slika 3.6 *Galera* grupna komunikacija 15

Slika 3.7 Implementacija *Galera cluste*ra zajedno s *keepalived* servisom na *HAproxy* poslužiteljima 16

Slika 3.8 Izgled konfiguracijske datoteke *Galera clustera* 18

Slika 3.9 Propuštanje *Galera cluster* kroz *SELinux* 19

Slika 3.10 Željeni ispis komande provjere nakon konfiguriranog *keealived* servisa 22

Slika 3.11 Spajanje na bazu podataka 24

Slika 4.1 Otkrivanje *hostova* u *subnetu* 25

Slika 4.2 Izgled *hosta* na centralnom sučelju s obaveznim parametrima 26

Slika 4.3Uspješno konfigurirano praćenje preko *agenta* 27

Slika 4.4 Princip rada aktivnog *agenta* 27

Slika 4.5 Princip rada pasivnog *agenta* 27

Slika 4.6 Sadržaj instalacijskog direktorija 28

Slika 4.7 Dodavanje SNMP sučelja na *hosta* 32

Slika 4.8 Definiranje *SNMP* šifre na *Zabbix* poslužitelju 32

Slika 4.9 Definiranje SNMP šifre na *Microsoft* poslužitelju 33

Slika 4.10 Uspješan indikator praćenja preko *SNMP-a* 33

Slika 4.11 Konfiguracija medija za pojedinog korisnika 34

Slika 4.12 Konfiguracija primanja upozorenja *gmailom* 35

Slika 4.13 Konfiguracija poslužitelja za slanje *mail* obavijesti preko skripte 37

Slika 4.14 Provjera uspješnosti slanja 37

Slika 4.15 Potvrda uspješnog slanja i prikaz izgleda poruke 39

Slika 4.16 Primjer definiranja *triggera* 41

Slika 4.17 Primjer grafičkog prikazivanja 42

Slika 4.18 *Zabbix Proxy* implementacija 44

Slika 4.19 Uspješna konfiguracija *Zabbix proxy* poslužitelja 46

Popis tablica

Tablica 2.1 Razine dostupnosti IT usluga 3

Popis kodova

Kod 3.1 Izgled *hosts* datoteke na svakom virtualnom poslužitelju 8

Kod 3.2 Parametri za promjenu u konfiguracijskoj datoteci 10

Kod 3.3 Propuštanje servisa kroz vatrozid 11

Kod 3.4 Uspješna autorizacija pcs servisa između poslužitelja 11

Kod 3.5 Propuštanje servisa kroz *SELinux* 14

Kod 3.6 Instalacija *MariaDB* poslužitelja koristeći *shell* skriptu 17

Kod 3.7 Izgled konfiguracijske datoteke galera *cluste*ra na db1 18

Kod 3.8 Redoslijed pokretanja *MariaDB* servisa i inicijalno pokretanja *clustera* 18

Kod 3.9 Propuštanje *MariaDB clustera* kroz *SELinux* 19

Kod 3.10 Kreiranje prazne baze podataka i dodavanje prava 20

Kod 3.11 Dodavanje inicijalne *sheme* za *zabbix* bazu podataka 20

Kod 3.12 Propuštanja kroz vatrozid za *Galera cluster* 20

Kod 3.13 Izgled *keepalived* konfiguracijske datoteke 22

Kod 3.14 Izgled *HAproxy* konfiguracijske datoteke 23

Kod 3.15 Konfiguracija skripte za aktivaciju servisa prilikom ponovnog pokretanja 24

Kod 3.16 Prebacivanje inicijalne *frontend* konfiguracije na drugi poslužitelj 24

Kod 4.1 *Batch* skripta za uređivanje konfiguracije *agenta* 29

Kod 4.2 *Powershell* funkcija za raspakiravanje arhive 29

Kod 4.3 Provjera i brisanje ako servis već postoji 29

Kod 4.4 Prebacivanje potrebnih i brisanje nepotrebnih datoteka s lokacije *agenta* 30

Kod 4.5 Pokretanje *batch* skripte za uređivanje konfiguracijske datoteke 30

Kod 4.6 Instalacija agenta i postavljanje automatskog pokretanja servisa 30

Kod 4.7 Propuštanje *agenta* kroz vatrozid i pokretanje servisa 30

Kod 4.8 Dodavanje repozitorija za *Zabbix agenta* 31

Kod 4.9 Uređivanje konfiguracijske datoteke na *CentOS* poslužitelju 31

Kod 4.10 Popuštanje sučelja kroz vatrozid 31

Kod 4.11 Konfiguracija slanja *mail* obavijesti preko skripte 36

Kod 4.12 Skripta za slanje *mail* obavijesti 36

Kod 4.13 Konfiguracija slanja SMS porukama 38

Kod 4.14 Komande za induciranje upozorenja 42

Kod 4.15 Dodavanje repozitorija za *Grafanu* 43

Kod 4.16 Skidanje paketa i pokretanje servisa 43

Kod 4.17 Integracija sa *Grafane* sa *Zabbixom* 43

Kod 4.18 Dodavanje repozitorija i skidanje paketa za *proxy* 45

Kod 4.19 Skidanje, instalacija i konfiguracija *proxy* baze podataka 45

Kod 4.20 Konfiguracija *proxy* servisa 46

Kod 4.21 Popuštanje kroz *SELinux* i vatrozid 46

Literatura

[1] ANDREA DALLEVACCHE; MASTERINGZABBIX; DECEMBER 2013;978-1-78328-349-1

[2] Patrik Uytterhoeven,ZabbixCookbook,March 2015;978-1-78439-758-6

[3] HTTPS://ERICSYSMIN.COM/2016/02/18/CONFIGURING-HIGH-AVAILABILITY-HA-ZABBIX-SERVER-ON-CENTOS-7/(PRISTUPANO 10. SIJEČNJA 2020.)

[4] HTTP://YALLALABS.COM/LINUX/HOW-TO-CONFIGURE-A-HIGH-AVAILABILITY-ZABBI-SERVER-USING-PACEMAKER-ON-CENTOS-7-RHEL-7/ (PRISTUPANO 10. SIJEČNJA 2020.)

[5] HTTPS://WWW.FOSSLINUX.COM/8328/HOW-TO-INSTALL-AND-CONFIGURE-GRAFANA-ON-CENTOS-7.HTM (PRISTUPANO 10. SIJEČNJA 2020.)

[6] HTTPS://ACCESS.REDHAT.COM/DOCUMENTATION/EN-US/RED_HAT_ENTERPRISE_LINUX/7/HTML/HIGH_AVAILABILITY_ADD-ON_OVERVIEW/S1-PACEMAKERARCHITECTURE-HAAO (PRISTUPANO 10. SIJEČNJA 2020.)

[7] HTTP://YALLALABS.COM/LINUX/HOW-TO-INSTALL-AND-CONFIGURE-ZABBIX-PROXY-ON-CENTOS-7-RHEL-7/ (PRISTUPANO 10. SIJEČNJA 2020.)

[8] HTTPS://WWW.ZABBIX.COM/DOCUMENTATION/4.0/MANUAL/INTRODUCTION (PRISTUPANO 10. SIJEČNJA 2020.)

[9] HTTPS://WWW.KEEPALIVED.ORG/(PRISTUPANO 10. SIJEČNJA 2020.)

[10] HTTPS://STACKOVERFLOW.COM/QUESTIONS/27768303/HOW-TO-UNZIP-A-FILE-IN-POWERSHELL(PRISTUPANO 10. SIJEČNJA 2020.)

[11] HTTPS://WWW.LOGICMONITOR.COM/BLOG/WHATS-WITH-THE-DIFFERENT-SNMP-VERSIONS-S1-V2C-V3/(PRISTUPANO 10. SIJEČNJA 2020.)

[12] HTTPS://CLUSTERLABS.ORG/PACEMAKER/DOC/EN-US/PACEMAKER/1.1/HTML/PACEMAKER_EXPLAINED/_PACEMAKER_ARCHITECTURE.HTML (PRISTUPANO 10. SIJEČNJA 2020.)

[13] HTTPS://WWW.PLESK.COM/BLOG/FEATURED/LINUX-LOGS-EXPLAINED/(PRISTUPANO 10. SIJEČNJA 2020.)

[14] HTTPS://WWW.HOWTOGEEK.COM/123646/HTG-EXPLAINS-WHAT-THE-WINDOWS-EVENT-VIEWER-IS-AND-HOW-YOU-CAN-USE-IT/(PRISTUPANO 10. SIJEČNJA 2020.)



ALGEBRA
VISOKO
UČILIŠTE

NASLOV ZAVRŠNOG RADA

Pristupnik: Frano Gatti, 0055482854

Mentor: Zlatan Morić