Tallinna Ülikool

Digitehnoloogiate Instituut

Kolmanda osapoole teenused rakenduse logimiseks

Seminaritöö

Autor: Robin Saar

Juhendaja: Jaagup Kippar

Autor: ……. „ ……..... „ 2016

Juhendaja: …….. „ ……..... „ 2016

Instituudi direktor: ……. „ ……..... „ 2016

Tallinn 2016

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Tänapäeva rakenduste puhul on kindlasti vaja pidada erinevaid logisid. Seda mitmel põhjusel. Esimene on kindlasti vigade logimine. Kui rakendus käitub mitte ettenähtud moel, siis on tarvidus näha, milles probleem on. Vigade kasutajale näitamine on äärmiselt halb praktika, seega on vaja need kokku koguda kusagile logi faili. Samuti on vaja ka koguda serverite siseseid vigu.

Teiseks põhjuseks on vajalike tegevuste logimine. Näitena võib tuua olukorra, kus klient rakendust kasutades ei saa soovitud tulemust. Kui teatud võtmetegevused on logitud, saab minna ja uurida, mis läks valesti ja miks. Korralikult logitud rakenduse puhul kajastub viga logides. See on kindlasti kiirem variant, kui hakata sama protsessi läbi tegema, mida klient tegi.

Suureks probleemiks peetakse logi failide hoidmist enda serverites või serveris, mida kasutab ka rakendus. Kõik arendajad on kunagi jõudnud olukorda, kus tekitatakse lõpmatu tsükkel. Kui see jääb aga mingitel põhjustel avastamata, jõuab serverisse ja antud tsüklis logitakse andmeid, siis võib logifaili kirjete arv suureneda ohjeldamatult. Seega kasvab ka faili maht. Kui jõutakse äärmusliku olukorrani, kus eelnevalt kirjeldatud olukord jääb pikalt avastamata, siis võib rakendus halvemal juhul täis kirjutada serveri andmemahud ja sellega kaasneb juba hulk muid probleeme.

Käesoleva seminaritöö eesmärgiks on tutvustada kolmanda osapoole teenuseid rakenduste logimiseks. Koolis, programmeerimise õppeainetes, tihtipeale ei räägita logimisest, või räägitakse pealiskaudselt. Samas on logimine rakenduse pikaealisuse tagamiseks ja haldamiseks tähtis tegevus. Aina enam hakatakse mõistma, et logide haldamine lokaalselt on mahukas ning seob endas väga suuri riske. Hilisem logide analüüsimine ja kokkuvõtete tegemine on aeganõudev ning seetõttu võetakse kasutusele kolmanda osapoole teenused, mis selle eest hoolitsevad. Teenusepakkujaid on palju, seega annab autor antud töös ülevaate enda poolt valitud kahest lahendusest. Need kaks teenusepakkujat sai valitud seetõttu, et nende puhul on kõik etapid põhjalikult dokumenteeritud.

Käesolev töö on jaotatud neljaks peatükiks. Esimeses peatükis tutvustab autor Logentriese poolt pakutavaid teenuseid. Lisaks ka erinevaid pakette ning võimalusi,

kuidas teenusesse on võimalik logisid saata.

Teises peatükis tutvustab autor Loggly poolt pakutavast teenusest. Samuti ka antud teenuse võimalustest ning protsessist, kuidas teenus andmeid töötleb.

Kolmandas peatükis loob autor mõlemasse teenusesse konto ning kasutab programmeerimiskeeles Python kirjutatud rakendust teenustesse esimeste logikirjete saatmiseks.

Neljandas peatükis kõrvutab autor eelnevalt tutvustatud lahendusi ning võrdleb nende teenuste võimalusi ja analüüsib, millisel juhul oleks teatud teenust parem kasutada.

1. Logentries

Logentries on Dr. Dublinis Viliam Holubi ja Dr. Trevor Parsonsi poolt loodud tarkvara ja teenuse pakkumise ettevõte (Techcrunch, 2012). Logentriese tehnoloogia võimaldab koguda ja analüüsida logisid, ning on üsnagi lihtsalt kasutatav. Lisaks on võimalik hiljem pärida kokkuvõtteid, statistikat ning seda kõike näha visuaalsel kujul. Kõik logid on olemas tsentraalsel kujul, ehk olenemata sellest, kus serveris, arvutis või domeenis rakendused on. (Logentries, 2016)

Ettevõte pakub mitmeid erinevaid pakette ning ka tasuta paketti väikeste mahtude tarvis. Tasuta paketi puhul on võimalik üles laadida 5GB infot kuus, ning andmeid talletatakse 7 päeva. Kõige suurema standardpaketi puhul on võimalik üles laadida kuni 150GB kirjeid ning seda talletatakse 30 päeva. Lisaks kõigele on võimalik nendega koos kokku panna ettevõtte jaoks kohandatud versioon. (Logentries, 2016)

1.1. Tähisega TCP

Token-based ehk tähisega logimine on üks Logentriese poolt välja pakutud logikirjete edastamise viis. Teenuses on võimalik ära määrata erinevad logide kogumikud, mis näiteks koosnevad erinevate rakenduste logidest, või ühe rakenduse alamosadest. Kui luua uus kogumik, on võimalik selle kogumiku jaoks luua uus ja unikaalne tähis, mille kuju on 2bfbea1e-10c3-4419-bdad-7e6435882e1f. Seda nimetatakse UUID (Universally unique identifier). Kasutades seda tähist logikirjes, saab teenus aru, millisesse kogumisse see hiljem paigutada. Kui näiteks on ühe rakenduse jaoks loodud kaks kogumikku, tuleb kirje saata kaks korda, erinevate tähistega. See lahendus sobib rakendustele, millest saadetakse logisid erinevatesse kogumikesse. Lisaks on see mõeldud sellistele rakendustele, millel võib muutuda IP aadress, ning sellisel juhul ei saa identifitseerimiseks kasutada IP aadressi. Selle lahenduse kasutamiseks on vaja teha TCP ühendus aadressil data.logentries.com, kasutades porti 80, 514 või 1000. Juhul kui kasutatav võrk ei ole usaldusväärne, on võimalik teha krüpteeritud ühendus samale aadressile, kuid kasutades porti 2000 (Logentries, 2016). Tähisega logide eelisteks on:

- saab lihtsalt saata suunatud logisid

- saab saata mitu logi rida, mitmest allikast ühele tähisega kogumikule
- saab lihtsalt luua ja organiseerida tähiseid iga logi kogumiku jaoks
- võimaldab laialdaselt oma logi struktuuri kohandada

1.2. TCP jaUDP

TCP ja UDP on teine Logentriese pool välja pakutud lahendus keskkonnaga suhtlemiseks. Selle lahenduse kasutamiseks tulen samuti luua TCP ühendus aadressil data.logentries.com, nagu ka eelmise lahenduse puhul. Erinevus seisneb selles, et antud ühenduse puhul määratakse keskkonnas igale ühendusele kindel pordi number. See määratakse, kui teenuses valida vastav logi saatmise moodus. Lahendus on siiski kasutatav vaid rakendustes, mille IP aadress ei muutu. Nimelt on IP aadressi ja prodi numbri kombinatsioon iga rakenduse puhul selle identifikaatoriks. Selle kombinatsiooni läbi saab teenus aru, kuhu kirje paigutada. See tähendab kas seda, et seda edastuse viisi saab kasutada juhul, kui logi kirjed saadetakse *Syslog daemon*i kaudu. Protsess algab tuvastusega, mille käigus oodatakse esimest TCP ühendust või esimest esimese UDP paketi saabumist. Antud lahenduse puhul on tuvastuse perioodi pikkus 15 minutit, mille jooksub tuleb teenusega ühendus saada. Seda nimelt turvalisuse tõttu, et tuvastus ei oleks avatud igavesti. Juhul, kui võrguühendus on ebaturvaline, saab kasutada ka krüpteeritud suhtlust. Selleks tuleb antud pordi numbrile liita juurde 10000 ja teha ühendus eelnevalt nimetatud aadressi asemel aadressile api.logentries.com (Logentries, 2016).

1.3. Erinevad sildid

Erinevaid logi ridu või sõnumeid saab sildistada. Selle peamine eelis tuleb välja hilisemal logida analüüsimisel. Kui eelnevalt on erinevad sõnumid sildistatud, siis hiljem saab neid siltide järgi otsida ja koondada. Lisaks saab määrata teatud siltide puhul, et nende esinemise korral antaks kliendile koheselt märku. Kui klient on määranud, et kõikidest vigadest, mis on sildistatud kui “error”, antaks märku, siis saab klient koheselt teada, kui mõnes rakenduses midagi valesti läks. Praegu saab kliente teavitada läbi järgnevate teenuste:

- Slack
- PagerDuty
- HipChat
- Campfire

- IPhone'le loodud rakendus
- Webhook.

Lisaks eelnevale on võimalus tellida märguandeid juhul, kui logides tekib mingisugune anomaalia. Näiteks kui logi parameetrid väljuvad kasutaja poolt ette seatud piiridest. Veel on võimalik saada teateid juhul, kui logidesse tekib teatud muster või logi muutub teatud ajaperioodiks tegevusetuks. Selleks, et passiivsuse teade liiga tihti ei korduks, on võimalus veel määrata, et sellest antakse teada ainult teatud arv tunnis või päevas. Seda selleks, kui passiivsuse määramise aeg on näiteks viis minutit, siis ei saadetaks teadet iga viie minuti tagant vaid korra tunnis või päevas näiteks.

Logentriese logisid saab kasutada ülesannete jagamiseks. Logile saab juurde panna meeskonna märkuse. See on hea funktsionaalsus, kui mõni rakendus on veel arendamise järgus. Märkustega saab logisid kommenteerida, saab määrata veateate mõnele arendajale või panna juurde, et probleem on töös või juba lahendatud (Logentries, 2016).

1.4. Datahub

Suurte rakenduste puhul, kus on kasutusel erinevad kasutajad ja nende autentimine, tuleb mõnikord maha logida ka salastatud infot. Nagu näiteks need samad isikuandmed, mis ei tohiks kunagi jõuda avalikku ruumi. Samas on vajalik ka seda infot hiljem analüüsida ja selle põhjal kokkuvõtteid teha. Logentries pakub selle probleemi lahendamiseks välja Datahub ja DataLock lahendused.

Datahub on keskne andmete hoida, mis on rakenduse tootja käes ja hallata. Kõigepealt jõuab logikirje Datahubi, kus määratud reeglite kohaselt krüpteeritakse salastamist vajav logi kirje osa. Peale krüpteerimist saadetakse logi Logentriese keskkonda. Keskkonda jõudnud logikirjes on salastatud info kohal genereeritud hash. Töötlemata informatsioon on endiselt talletatud kohalikku DataHubi ning vajadusel saavad volitatud isikud neid andmeid töödelda. (Logentries, 2016)

DataLock on eelnevaga kaasnev laiendus, millega saab määrata isikud, kellel on õigus näha filtreerimata informatsiooni. Samuti vastutab see laiendus selle eest, et kui volitatud kasutaja

Logentriese keskkonnas hakkab kirjeid analüüsima, siis tema näeb salastatud infot. Laiendus saab aru, et tegemist on selle kasutajaga, ning keskkonnas oleva info ja kohalikus DataHubis olev info sünkroniseeritakse. (Logentries, 2016)

Kuigi valik informatsiooni krüpteeritakse, ei tähenda see, et ülejäänud seda infot ei saaks analüüsida või töödelda. Endiselt saab teha Logentriese poolt pakutavaid funktsioone, kuid seda siis juba krüpteeritud väärtustega. Seega ei tähenda info mitte nägemine seda, et inimene, kes logisid analüüsib, ei saaks teatud hulka kirjeid töödelda.

2. Loggly

Loggly on logide analüüsimiseks ja käitlemiseks loodud pilveteenus, mis sai alguse 2009 aastal. Tegemist on kolme mehe poolt loodud ettevõttega, mille mõte oli lahendada ettevõtete operatiivsed probleemid. 2013 aastal anti välja uuendatud versiooni oma algsest lahendusest, mis võimaldas koguda logisid syslogi kaudu ning pakuti võimalust kasutada graafilist kasutajaliidest. Ettevõtte põhisõnum on, et DevOps, SysOps ja teised insenerid ei peaks muretsema logide pärast muretsema ja neid analüüsima.

Nagu ka Logentries, pakub Loggly erinevaid pakette ning ka tasuta paketti, mis jääb igavesti tasuta. Tasuta paketi puhul on nad valmis võtma 200MB päevas ja talletama seda 7 päeva. Kõige suurem pakett võimaldab saata kuni 75GB andmeid päevas ja neid talletada üle viieteistkümne päeva. Samuti pakub ka Loggly võimalust kokku leppida erilahendusi, kus andmemahud võivad olla veel suuremad ja talletamise periood veel pikem. (Loggly, 2016)

2.1. Andmete kogumine

Loggly kasutab andmete kogumiseks juba olemasolevaid ja avatud standardeid, nagu näiteks syslog ja HTTP. Seda selleks, et klient ei peaks installima suurt tarkvara igasse arvutisse. Arvestades, et *native*-lahendusega rakendus võib olla tuhandetes seadmetes, ei ole logide kogumiseks lisarakenduse installimine mõtekas. Veebipõhiste rakenduste puhul ei oleks probleem niivõrd suur. Samuti pole vahet, mis keeles rakendus on kirjutatud. Kui rakendusest on võimalik logida, siis on seda võimalik ka Logglysse logida (Loggly, 2016). Kui vahetevahel logidakse rohkem kui ostetud paketis kirjas, siis teenus selle peale ei pahanda ja logid jõuavad ikkagi keskkonda. Nad saavad aru, et mõni rakendus võib muutuda hetkeks kontrollimatuks või uue lahenduse testimisel võib logisid rohkem tekkida (Loggly, 2016).

2.2. Andmete vastu võtmine, kirjete analüüs ja otsing

Loggly ise ütleb andmete vastuvõtmise kohta, et siin neelatakse andmeid. Selles protsessis saab logi kirjetesse lisada märkusi ja muud metadatat. Teenus võimaldab algsetele logikirjetele lisada tuletatud väljasid, mis tähendab, et enne andmete talletamist suudetakse

juba teha arvutusi ja kokkuvõtteid. Siin faasis tehtud töö tulemus on näha hilisemas analüüsimise järgus.

Kirjete analüüsimisel pakutakse reaalajas analüüsitud andmeid. Iga kord, kui logikirje lisatakse, arvutab ja analüüsib keskkond selle info läbi ja pakub seeläbi võimalust kiirelt reageerida sündmustele. Lisaks paigutab Loggly väga kiirelt kirjed ka kataloogidesse ning seeläbi on andmed sisuliselt koheselt kättesaadavad ning kasutatavad. Selleks, et kogu asi kiirelt ja sujuvalt toimiks, on Loggly välja töötanud *Dynamic Field Explorer*i. Sisuliselt tähendab see, et analüüs ei alga kunagi tühjalt lehet. Keskkond pakub kasutajatele juba eelnevalt summeeritud tulemusi, logikirjete struktuuri ja määratud märksõnade esinemise sagedust. See kõik toimub keskkonna poolt loodud intelligentse algoritmi läbi, mis eeldab, mida kasutajad näha soovivad. Näidatakse ülevaadet kõge sagedamatest juhtumitest kui ka anomaaliatest ning läbi selle saab jõuda väga kiirelt ka mõne kindla kirjeni. Peamiselt on Loggly loonud selle võimaluse, kuna kasutajad tihtipeale ei tea, millisest kohast andmeid analüüsima hakata. Summeeritud andmeid pakutakse kohe seetõttu, et kasutaja ei peaks ise neid liitma ja filtreerima hakkama ning lisaks näidatakse kasutajate koheselt määratud kirjete tüüpide esinemise sagedust. Tihti on peamine teada saada näiteks vigade esinemist või teatud tüüpi vea esinemist, siis teenus analüüsib need andmed eelnevalt ära, et kasutajal oleks koheselt teada, kas viga on esinenud. (Loggly, 2016)

2.3. Monitooring, reaalajas häired ja anomaaliate avastamine

Loggly keskkonnas on peamine rõhk pandud just sellele osale. Seda selleks, et andmete kogumine ei oleks tulutu tegevus. Igat päringut ja kogutud andmeid saab muuta graafikuteks ning jälgida reaalajas visuaalselt. Samuti on graafikuid võimalik asetada üksteise peale, mis loob võimaluse korraga hallata suurt info mahtu. Lisaks kuvatakse kasutajale reaalajas häireid. Teenusesse saab kindlaks määrata, millistel juhtudel kuvada häiret. Häire puhul saab paika panna teatud logikirje tüübi esinemist, kindlad künnised, mille ületamisel antakse haldajale märku või muude parameetrite järgi paika pandud piirid, millest väljumisel tuleb teavitada. Häireid kuvatakse muidugi Loggly keskkonnas endas, kuid lisaks on võimalik seadistada häirete kuvamine muudesse teenustesse.

Loggly pakub võimalust saata häired järgmistesse teenustesse:

- HipChat
- PagerDuty
- Slack

Loggly on oma teenusesse sisse ehitanud iseõppiva algoritmi, mis õpib logide normaalset mustrit (Loggly, 2016). Võib juhtuda olukord, kus tekivad logikirjed, mis ei vasta mustrile ja ei ole määratud ka häirena. Selleks näitab keskkond koheselt haldajale ka logisid, mis ei vasta antud mustrile ning tänu millele võib kasutaja varakult märgata tekkivat viga. Kui näiteks on tekkinud muster, et andmeid sünkroniseeritakse mingi aja tagant API pihta ning järsku seda enam ei toimu, saab haldur tekkinud anomaaliast kiirelt aru. Mõndade rakenduste puhul, nagu näiteks kassasüsteemid või veebipoed, võib andmete sünkroniseerimine olla väga tähtis, kuna need sünkroniseerivad näiteks laoseisusid API pihta. Anomaaliate tuvastamine on üldiselt ikka selleks, et kasutaja saaks võimalikult ruttu läbi logide teada, kus on tekkinud probleemid rakenduses.

Logikirjed on struktureeritud ja jagatud kataloogidesse, seega on võimalik ka määrata kasutajate grupp, kes teatud kirjeid näeb ja kellel häire või anomaalia kuvatakse. Kuna logidesse kirjutatakse nii serveri infot, kui ka rakenduse infot, siis ei ole mõistlik kuvada serveri haldajale rakenduses tekkinud probleeme ja vastupidi. Vastasel juhul tekiks palju müra ning analüüs oleks raskendatud ja mitte kontsentreeritud. (Loggly, 2016)

3. Teenuste kasutamine

Käesolevas peatükis näitab autor kuidas seada üles Python rakendus, mis hakkab mõtlemasse eelpool nimetatud keskkondadesse logi kirjeid saatma. UNIX operatsioonisüsteemides on Pythoni rakenduse loomine ja sellesse komponendi lisamine tehtud väga lihtsaks. Kui eelnevalt pole installeeritud Pythonit, siis tuleb seda kindlasti teha. Selleks tuleb allalaadida vastav pakk ja käivitada installer või teise moodusena kogu protsess läbi teha käsureal.

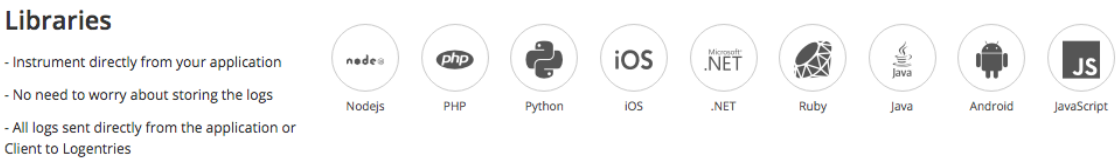
Logentries ja Loggly on oma paketi üles laadinud ka Pythoni paketihaldurisse pip. Pip on Pythoni poolt soovitatud tööriist Pythoni pakettide installimiseks.
Pip installeerimine erinevates UNIX laadsetes operatisoonisüsteemides:

- pip installeerimine Mac OS X
  $ sudo easy_install pip
- pip installeerimine Ubuntu, Debian or Linux Mint:
  $ sudo apt-get install python-pip
- pip installeerimine Fedora:
  $ sudo yum install python-pip
- pip installeerimine CentOS, first enable EPEL repository, and then run:
  $ sudo yum install python-pip
- pip installeerimine Archlinux:
  $ sudo pacman -S python-pip
- pip installeerimine Windows
  $ *python get-pip.py*

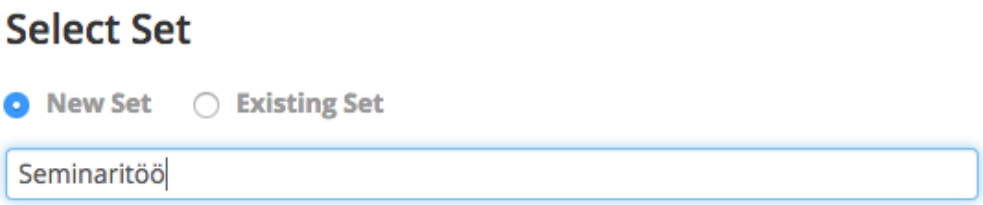
3.1. Logentries

3.1.1. Konto loomine

Logi kirjete saatmise alustamiseks tuleb esmalt luua kasutajakonto, kui see juba eelnevalt registreeritud pole. Kui konto saab registreeritud tuleb valida teenuses vastava programmeerimiskeele *library* (Pilt 1). Seejärel saab luua uue logi kogumiku omalt poolt valitud nimega (Pilt 2).



Pilt 1. Keelte valikud



Pilt 2. Logi kogumiku loomine

Käesolevas töös kasutab autor näidete loomisel Token TCP võimalusi. Selleks, et teenusesse saaks logikirjeid saata nõuab Logentries unikaalset tähist. Nagu eelnevalt öeldud, suudab teenus selle järgi eristada erinevaid kirjeid, rakendusi ja kasutajaid. Tähis genereeritakse automaatselt teenuse poolt, ning see tuleb lisada oma rakenduse koodi.

3.1.2. Rakenduse loomine ja logide saatmine

Kui on installeeritud pip, tuleb kasutada seda paketihaldurit, et installeerida Logentriese pakett, kasutades käsurea käsku '*$ sudo pip install logentries*'. Seda tuleb teha superkasutaja rollis, kuna vastasel juhul ei pruugi kasutaja olla õigusi, et muuta Pythoni kataloogi sisu. Alla laetud paketi kasutamiseks rakenduses tuleb see importida, lisaks muudele vajalikele komponentidele (Koodinäide 1).

```
from logentries import LogentriesHandler
import logging
import time
```

Koodinäide 1. Vajalike komponentide importimine

Esimeste logikirjete saatmiseks Logentriese teenusesse ei pea palju vaev nägema. Tuleb *logging* komponenti kasutades luua uus logger ja panna selle nimetuseks 'logentries'. Kuna kasutatav meetod tagastab objekti, tuleb see panna muutujasse. Seejärel saab määrata logi taseme ja lisada *handleri* koos Logentriese keskkonnas antud tähisega (Koodinäide 2).

```
log = logging.getLogger('logentries')
log.setLevel(logging.INFO)
log.addHandler(LogentriesHandler('Insert Logentries token here'))
```

Koodinäide 2. Loggeri seadistamine

Eelneva tegevuse tagajärjel on nüüd võimalik saata esimesed logikirjed teenusesse. Kuna loodud rakendusel puudub sisu ning on loodud vaid selleks, et demonstreerida logide saatmist teenusesse, siis on vajalik lisadalogide saatmise read. Kasutades eelnevalt muutujasse salvestatud objekti, tuleb välja kutsuda logi taseme meetod. Kuna teenust pakub võimalust kasutada erinevaid kirjete tasemeid, siis tuleb vastav meetod ka kasutusele võtta, et keskkond saaks aru, mis laadi kirjega on tegemist.

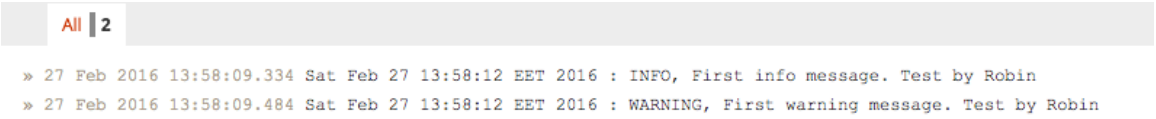
Logi kirjete tasemed on:

- debug
- info
- notice
- warning
- error
- critical
- alert
- emerg

```
log.info('First info message. Test by Robin')
log.warn('First warning message. Test by Robin')
```

Koodinäide 3. Esimeste logikirjete saatmine

Kasutades koodinäites näidatud käsklusi saadetakse teenusesse esimesed kirjed (Koodinäide 3). Kirjeid saadetakse teenusesse asünkroonselt. See tähendab, et rakendus ei jää ootama vastust või tulemust sellele käsklusele, vaid läheb oma protsessidega edasi. Selle tulemusena saab rakendus töötada omaette ja ei olene välistest tõrgetest. Edu korral on kirjed keskkonnas nähtavad (Pilt 3).



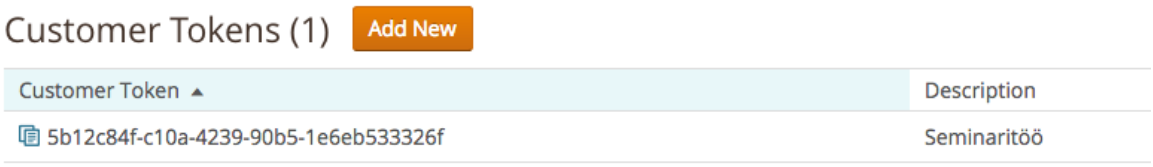
Pilt 3. Esimesed kirjed teenuses

3.2.Loggly

3.2.1. Konto loomine

Teenuse kasutamiseks tuleb esmalt luua teenusesse konto, kui see juba eelnevalt loodud pole. Kui konto loodud, tuleb valida programmeerimiskeelele vastav teek(Pilt 4). Seejärel pakub Loggly esmalt võimalust saata logid läbi syslogi. Käesolevas töös kasutab autor näidete loomiseks tähisega logide saatmise võimalust. Selline valik on teenuse menüüribal olemas, nimega 'Customer Tokens' (Pilt 5).

Pilt 4. Keelte valikud



Pilt 5. Loodud tähis

3.2.2. Rakenduse loomine ja logide saatmine

Kui on installitud pip, saab Loggly paketi haldurist kätte käsuga '*$ sudo pip install loggly-python-handler*'. Käsku tuleb käsureal käivitada superkasutaja õigustes, kuna vastasel juhul ei pruugi olla õigusi Python kataloogi muutmiseks.

Selleks, et rakendus saaks teenusesse logisid saatma hakata, tuleb luua konfiguratsioonifail (Koodinäide 5) ja see rakenduses importida (Koodinäide 4).

```
import logging
import logging.config
import loggly.handlers

logging.config.fileConfig('python.conf')
```

Koodinäide 4. Komponentide importimine

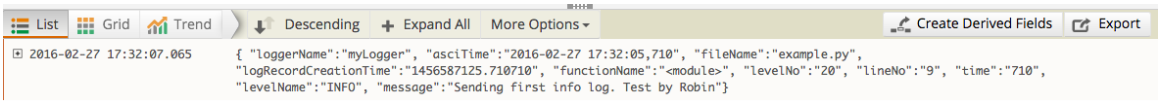
```
[handlers]
keys=HTTPSHandler

[handler_HTTPSHandler]
class=loggly.handlers.HTTPSHandler
formatter=jsonFormat
args=('https://logs-01.loggly.com/inputs/CUSTOM_TOKEN_HERE/tag/python','POST')

[formatters]
keys=jsonFormat

[loggers]
keys=root

[logger_root]
handlers=HTTPSHandler
level=INFO

[formatter_jsonFormat]
format={ "loggerName":"%(name)s", "asciTime":"%(asctime)s", "fileName":"%(filename)s", "logRecordCreationTime":"%(created)f",
"functionName":"%(funcName)s", "levelNo":"%(levelno)s", "lineNo":"%(lineno)d", "time":"%(msecs)d", "levelName":"%(levelname)s",
"message":"%(message)s"}
datefmt=
```

Koodinäide 5. Konfiguratsiooni faili näide

Selleks, et imporditud komponenti kasutata tuleb luua muutuja, kuhu salvestada tekkiv objekt vabalt valitud nimega. Seejärel saab välja kutsuda objekti meetodi, logi kirje saatmiseks (Koodinäide 6). Nagu ka Logentriese puhul on võimalik kasutada erinevaid logi kirjete tasemeid. Esimese näitena saadetakse infokirje. Sellega saab testida, kas siiani tehtu töötab ja kirje jõuab teenusesse (Pilt 6).

```
logger = logging.getLogger('myLogger')
logger.info('Sending first info log. Test by Robin')
```

Koodinäide 6. Esimese kirje saatmine



Pilt 6. Esimesed logi kirjed teenuses

4. Analüüs

Käesolevas töös näitas autor, kuivõrd lihtne ja sirgjooneline on rakenduste logide saatmine kolmanda osapoole teenustesse. Vajalik on vaid mõningate eelduste täitmine ning esimesed logikirjed jõudsid teenuste keskkonda. Kindlasti ei ole antud töös näidatud lahendused lõplikud ja sobilikud kõikide rakenduste ja lahenduste jaoks. Juhul, kui rakendus on suurem, ning kasutatav paljude klientide poolt, võivad käesolevas töös näidatud lahendused olla ebaturvalised ja nõuda liigselt haldust rakenduste skaleerumisel.

Mõlemad töös tutvustatud teenusepakkujad on vaeva näinud, et nende poolt pakutav teenus oleks võimalikult lihtsalt rakendusse integreeritav. Käesolevas töös oli skoop väike ning seetõttu ka integratsioon lihtne. Mõlemate teenusepakkujate teenuse rakendamine oli piisavalt lihtne ning ei nõudnud palju eelnevat teadmist. Vaja oli operatsioonisüsteemis vajalike komponentide installeerimine ning nende kasutamine. Peale seda tehti palju tööd arendaja eest ära komponentide poolt.

Võrreldes kahte teenust, arvab autor, et Logentriese poolt pakutud lahendus oli lihtsam. Seda nimelt seetõttu, et ei olnud vajadust konfiguratsiooni faili järele. Sai importida komponendi, ning see tegi oma töö väheste parameetritega ära. Loggly puhul vajaminev konfiguratsiooni fail ei olnud keeruline ning kahtlemata tuleb selle faili täielik potentsiaal välja suuremate rakenduste puhul.

Keskkondadesse jõudnud logikirjed olid samuti Logentriese poolel lihtsamal kujul ning lakoonilisemad. Teenusesse jõudis aeg, kirje tase ning sõnum. Sellest enamjaolt piisab ning see ei tekita liigset müra logi kirjetes. Loggly puhul teenusesse jõudnud logi kirje oli palju keerukam. Samuti olid esindatud aeg, kirje tase ja sõnum, kuid lisaks saadeti ka failinimi, taseme number ja koodirea number. Kindlasti tuleb Loggly poolt talletatud logikirje kasuks juhul, kui rakendus on suurem ja faile rohkem.

Peale mõlema teenusepakkuja kasutamist arvab autor, et Loggly on sobilik peamiselt suurte rakenduse puhul. See ei tähenda muidugi, et väikeste rakenduste puhul seda kasutada ei tohiks, kuid on näha, et teenusepakkuja on oma teenuse üles ehitanud suurte rakenduste teenindamiseks. Samuti ei tähenda see seda, et Logentries ei oleks sobilik suurte rakenduste

puhul. Aga käesoleva seminaritöö skoopi arvestades oli autor meelest sobilikum ja lihtsam kasutada Logentriese teenust.

Kokkuvõte

Käesolevas seminaritöös tutvustatud teenused ja nende kasutamise võimalused on autori arvates vaid lühike ja pinnapealne kokkuvõte. Kindlasti saab sügavamalt süveneda teistesse teenustesse ja nende võimalustesse. Samuti ka tutvustatud teenuste teistesse lahendustesse. Antud seminariöö eesmärgiks oli tutvustada rakenduste logimist ja kasutada logide hoidmiseks kolmanda osapoole teenuseid. Eesmärgi saavutamiseks võeti kasutusele kaks teenusepakkujat, Logentries ja Loggly. Autor lõi väikesed rakendused, eesmärgiga saata esimesed logikirjed teenustesse ning näidata selleks kõige lihtsamat moodust ja vajalikku ülesse seadmise protsessi.

Teenuste põhimõte ja vajadus on hoida rakendused ja nende logid eraldi. Seda nii turvakaalutlustel, kui ka hilisema logide analüüsi hõlbustamiseks. Logide kirjutamine ja nende haldus on hetkel Tallinna Ülikooli programmeerimise õppeainetes nõrgalt esindatud ning autor arvab, et käesolev töö võib olla õppematerjal järgnevatele kursustele.

Autori hinnangul sai käesoleva töö eesmärk saavutatud, kuna töös jõuti logikirjete teenusesse saatmiseni, ning anti ülevaade kahest teenusepakkujast ja nende poolt pakutavatest võimalustest. Autor ise sai teadmisi ja kogemusi logide olemusest ning nende saatmisest teenusepakkujate keskkond. Lisaks veel teadmised vajalikest komponentidest ning paketihaldurist pip.

Autor arvab, et antud teemal on võimalik edasi uurida, kuidas teenusepakkuja poolt pakutavad lahendused töötavad suurema rakenduse puhul ning kuidas võimaldavad teenusepakkujad hilisemat logide analüüsi ja haldamist.

Kasutatud kirjandus

Logentries (2016) Pricing. Loetud 27.02.2016 aadressil:

https://logentries.com/pricing/

Logentries (2016) Token-based TCP. Loetud 27.02.2016 aadressil:

https://logentries.com/doc/input-token/

Logentries (2016) Syslog Forwarding. Loetud 27.02.2016 aadressil:

https://logentries.com/doc/syslog/

Logentries (2016) Tags and Alerts. Loetud 27.02.2016 aadressil:

https://logentries.com/doc/setup-tags-alerts/

Logentries (2016) Datahub. Loetud 27.02.2016 aadressil:

https://logentries.com/product/datahub/

Loggly (2016) Pricing. Loetud 27.02.2016 aadressil:

https://www.loggly.com/plans-and-pricing/

Loggly (2016) Product. Loetud 27.02.2016 aadressil:

https://www.loggly.com/product/

Loggly (2016) Dynamic Field Explorer. Loetud 27.02.2016 aadressil:

https://www.loggly.com/docs/dynamic-field-explorer/

Npmjs (2016) Node-logentries. Loetud 27.02.2016 aadressil:

https://www.npmjs.com/package/node-logentries

Python (2016) pip 8.0.3. Loetud 27.02.2016 aadressil:

https://pypi.python.org/pypi/pip

Techcrunch (2016) Logentries. Loetud 27.02.2016 aadressi:

http://techcrunch.com/2012/07/17/logentries-raises-1m-to-join-the-big-data-log-management-crowd/