TAMPEREEN
AMMATTIKORKEAKOULU

MACHINE LEARNING JA DATA SCIENCE - RATKAISUT OPENSHIFTISSÄ

Vilho Stenman

Opinnäytetyö
Marraskuu 2018
Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

STENMAN, VILHO:
Machine learning ja data science -ratkaisut OpenShiftissä

Opinnäytetyö 33 sivua
Joulukuu 2018

Opinnäytetyössä selvitettiin, kuinka saada data science ja machine learning toteutuksia ajettua OpenShiftin päällä mahdollisimman helposti. OpenShiftiä oli alunperin tarkoitus käyttää toimeksiantajan ympäristössä, mutta opinnäytetyön aikana ympäristöksi vaihdettiin lokaalisti asennettava OpenShift. Ympäristönvaihdos suoritettiin useiden eri machine learning ja data science -ratkaisuiden asennusten epäonnistuttua OpenShiftin käyttöoikeuksien vuoksi. Opinnäytetyön toimeksiantaja on Ambientia Oy, joka toimii ohjelmistoalalla räätälöityjen ohjelmistotopalveluiden toteuttajana, sekä konsultointipalveluiden ja Atlassian-lisenssien ja -palveluiden toimittajana.

Opinnäytetyössä selvitettiin, kuinka TensorFlowin ja Apache Sparkin asentaminen ja käyttöönotto tapahtuu OpenShiftissä. TensorFlow ja Apache Spark ovat avoimen lähdekoodin koneoppimis- ja datatiedesovelluksia. Opinnäytetyössä tutkittiin, kuinka kyseiset ratkaisut toimivat OpenShiftissä ja pohdittiin sekä vertailtiin näiden ratkaisuiden ominaisuuksia ja toiminnallisuuksia. Opinnäytetyössä pohdittiin myös infrastruktuurin perustamisen vaivaa ja yksityisen pilven skaalautuvuutta verrattuna julkisen pilven rajattomaan skaalautuvuuteen.

Koneoppiminen ja datatiede kasvavat tällä hetkellä nopeasti ja valtaavat lisää markkinaosuutta. Opinnäytetyössä perehdyttiin tarkemmin siihen, mitä koneoppiminen ja datatiede ovat ja mitä kaikkea niillä voidaan tehdä ja saada aikaan.

Asiasanat: koneoppiminen, datatiede, openshift, tensorflow, apache spark

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Software Production

STENMAN, VILHO:
Machine Learning and Data Science in OpenShift

Bachelor's thesis 33 pages
December 2018

The objective of the thesis was to find out how to get machine learning and data science implementations on OpenShift with a relative ease. The client's OpenShift was intended to be used as the environment but during the thesis the environment was changed to a local OpenShift installation. The environment change was carried out due to the failure to install several machine learning and data science solutions because of OpenShift´s limited permissions. The thesis was commissioned by Ambientia Oy, a company that is providing tailored software services and is a supplier of consultancy services and Atlassian licenses and services.

Research was done on how to set up and deploy TensorFlow and Apache Spark on OpenShift. Research was also done on how these implementations work in OpenShift and how the features and functionalities of the implementations compare to each other. The thesis also focused on the difficulty of creating an infrastructure and the scalability of a private cloud compared to the unlimited scalability of the public cloud.

Machine learning and data science are both rapidly growing and their market share is accelerating. The thesis focused on what machine learning and data science are and what can be done and achieved by using them properly.

Key words: data science, openshift, machine learning, tensorflow, apache spark

SISÄLLYS

1 JOHDANTO JA TYÖN LÄHTÖKOHDAT ... 6
1.1 Toimeksiantaja ... 6
1.2 Tavoitteet ... 6
2 PILVIPALVELUT, BIG DATAN KÄYTTÖ JA HYÖDYT ... 7
2.1 Johdanto ... 7
2.2 Data science ... 7
2.3 Machine learning ... 8
2.4 Saas, paas, iaas ja niiden kehitys ... 10
2.5 Openshift, kontit, Platform as a service ... 12
2.6 Koneoppimisen ja datatieteen hyödyt Openshiftissä ... 14
3 KONEOPPIMISEN JA DATATIETEEN RATKAISUT ... 15
3.1 Johdanto ... 15
3.2 Datatiede ja koneoppimis -ketjun kuvaus ... 15
3.3 TensorFlow ... 16
3.4 TensorFlowin asennus ja käyttöönotto ... 16
3.5 Apache Spark ... 18
3.6 Apache Sparkin asennus ja käyttöönotto ... 20
3.7 Mahdolliset muut ratkaisut ja niiden läpikäynti ... 23
4 RATKAISUIDEN AJAMINEN OPENSHIFTISSÄ ... 24
4.1 Ratkaisuiden vertailu ... 24
4.2 Yhteenveto ... 25
5 SKAALAUTUVUUS JA INFRASTRUKTUURIN PERUSTAMINEN ... 26
5.1 Skaalautuvuus yksityisessä pilvessä verrattuna julkiseen pilveen ... 26
5.2 Infrastruktuurin perustamisen vaiva ... 28
6 POHDINTA JA JOHTOPÄÄTÖKSET ... 29
LÄHTEET ... 31

LYHENTEET JA TERMIT

OpenShift	Red Hatin tarjoama PaaS
PaaS	Platform as a Service eli palvelualustan ulkoistaminen
URL	Uniform Resource Locator eli verkkosivun osoite
machine learning	koneoppiminen
data science	datatiede
big data	yhteisnimitys valtaisille datamäärille
server	palvelin
template	OpenShiftissä käytettävä sovellussapluuna
AWS	Amazon Web Services eli Amazonin pilvipalvelualusta
oc	OpenShift CLI
CLI	command line interface eli komentokehote
CDK	Red Hat Container Development Kit
API	Application Programming Interface eli ohjelmointirajapinta

1 JOHDANTO JA TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii suomalainen vuonna 1996 perustettu ohjelmistoyritys Ambientia Oy, joka toimii ohjelmistoalalla räätälöityjen ohjelmistotopalveluiden toteuttajana, sekä konsultointipalveluiden ja Atlassian-lisenssien ja -palveluiden toimittajana. Ambientialla on toimistoja Suomessa viidellä eri paikkakunnalla ja kaksi toimistoa myös Virossa. Tällä hetkellä työntekijöitä on noin 170. Ambientialla OpenShift on ollut käytössä jo parin vuoden ajan ja sitä yritetään saada yhä useammille Ambientian asiakkaille käyttöön.

1.2 Tavoitteet

Opinnäytetyön tavoitteena on selvittää, miten saada data science eli datatieteen ja machine learning eli koneoppimisen -toteutuksia ajettua OpenShiftin päällä (sovelluskehittäjän näkökulmasta) mahdollisimman helposti. Tarkoituksena siis on ottaa selvää erilaisista koneoppimisen ja datatieteen toteutuksista, ja selvittää, kuinka kunkin ajaminen onnistuu OpenShiftiin. Ambientian intressinä olisi saada OpenShiftiin samaa toiminnallisuutta, mitä esimerkiksi Google Cloud Platform jo tällä hetkellä tarjoaa. Opinnäytetyössä tulee tuoda myös esiin näkökulma skaalautuvuudesta yksityisessä pilvessä verrattuna rajattomaan skaalautumiseen julkisessa pilvessä, sekä koko infrastruktuurin perustamisen vaiva verrattuna siihen, että otettaisiin käyttöön jotain sellaista, mitä monet osaajat ovat jo koostaneet. Opinnäytetyössä olisi myös hyvä miettiä, kuinka koneoppimisen ja datatieteen -ratkaisuja OpenShiftin päällä voitaisiin mahdollisesti myös tuotteistaa.

2 PILVIPALVELUT, BIG DATAN KÄYTTÖ JA HYÖDYT

2.1 Johdanto

Tässä luvussa käydään läpi keskeisiä käsitteitä ja pohditaan myös koneoppimisen ja datatieteen hyötyjä OpenShiftissä. Pilvipalveluiden suosio on viimeisinä vuosina lähtenyt räjähdysmäiseen kasvuun. Youtube, Spotify ja Netflix ovat jokaiselle tuttuja pilvipalveluja, joiden tarjoamaan sisältöön pääsee käsiksi lähes mistä ja milloin tahansa, esimerkiksi kannettavan tietokoneen tai mobiililaitteen avulla. Pilvipalvelussa tiedostot ja ohjelmat ovat pilvessä eli pilvipalveluntarjoajan palvelimella, eivätkä käyttäjän omalla koneella. Konkreettisella tasolla pilvipalvelu siis tarkoittaa tietokoneiden eli palvelimien verkostoa.

Perinteisesti yritysten omat ohjelmat, verkkosivut ja tiedostot ovat olleet yritysten omissa tiloissa tai yritysten vuokraamassa palvelinkeskuksessa. Pilvipalveluiden yleistyttyä yhä useammat yritykset ovat ottaneet käyttöön pilvipalveluita tarjoavan yrityksen kautta palvelimia käyttöön ja yritysten omat niin sanotut server roomit eli palvelinhuoneet ovat vähentyneet. Opinnäytetyössä käytettävä OpenShift ei suoranaisesti ole pilvipalvelu, vaan PaaS, eli Platform as a Service, mutta samalla myös yksityinen pilvi. OpenShift voi kuitenkin asennuksesta riippuen olla pilvipalvelu tai paikallinen asennus.

2.2 Data science

Nykyään dataa kertyy kaikkialta: tietokoneista, kameroista, älypuhelimista, jokaisesta kuviteltavasta sensorista, aktiivisuusrannekkeista ja älykelloista. Jokaisesta sosiaalisen median päivityksestä, hakukonehausta, aktiivisuusrannekkeen tai älypuhelimen kanssa tehdystä juoksulenkistä kertyy dataa. Kun dataa on niin paljon, että perinteisten tietokantakyselyiden palauttaman datan käsittely ja määrä aiheuttaa ongelmia, käytetään siitä käsitettä big data. Pelkällä datalla ei tehdä mitään, ellei sitä käsitellä jollain tavalla ja generoida siitä jotain hyödyllistä, mistä taas muodostuu termi datatiede. Datatieteen avulla voidaan tuottaa datasta graafeja, tilastoja ja ennustuksia, joiden avulla voidaan paremmin ymmärtää ja parantaa esimerkiksi liiketoimintaa. (Pierson & Porway. 2017, 7-10.)

Datatiede saatetaan helposti sekoittaa datan prosessointiin, data engineeringiin tai muuhun datan käsittelyyn liittyvään. Vaikka nämä ovat datatieteen kannalta olennaisia ja tärkeitä osa-alueita, ovat ne kuitenkin huomattavasti yleisempiä, kuin itse datatiede ja on tärkeää ymmärtää niiden eroavaisuudet. Datatieteessä tulee päästä käsiksi dataan ja siinä hyödynnetään datan prosessointia ja data engineeringiä. Ne tukevat datatiedettä, mutta ovat hyödyllisiä muussakin, kuten modernin webjärjestelmän prosessoinnissa tai onlinemainoskampanjan käsittelyssä. (Provost & Fawcett. 2013, 7-8.)

2.3 Machine learning

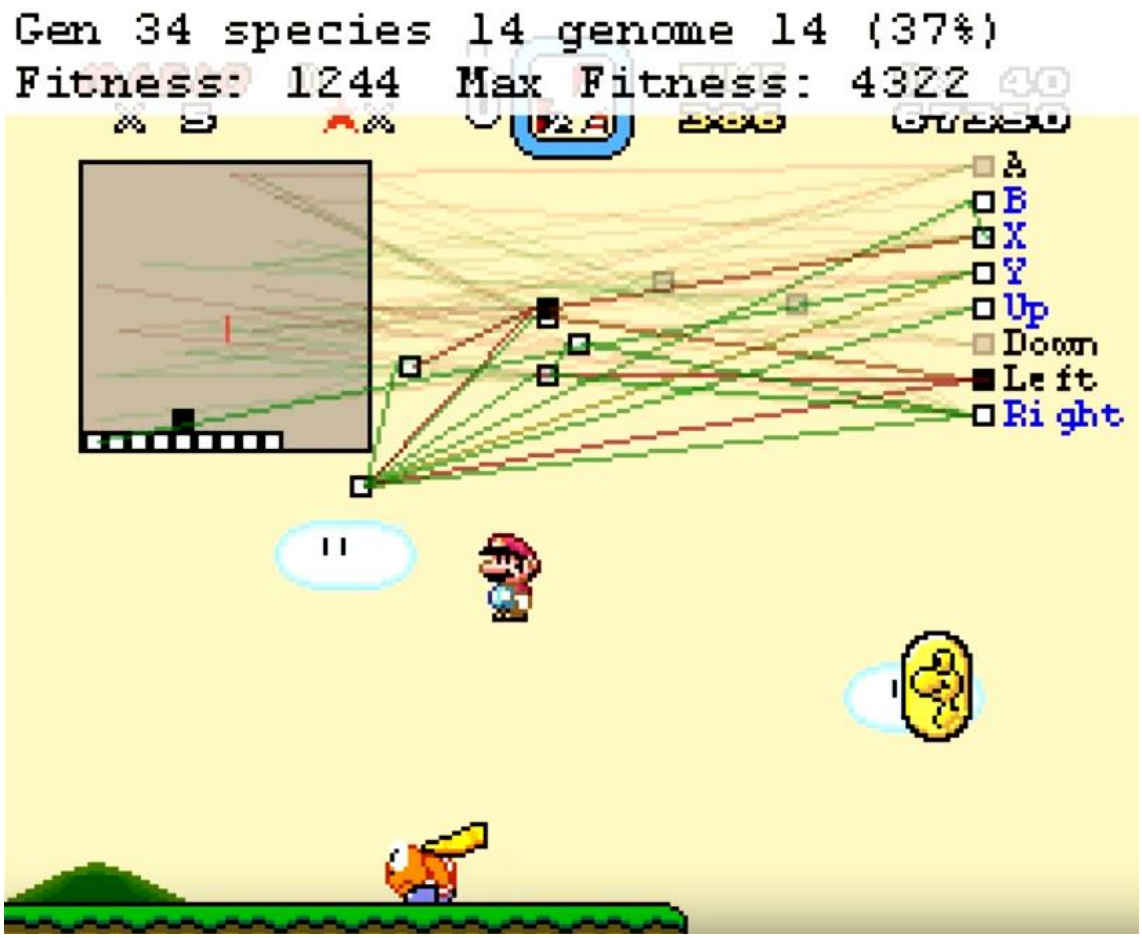
Machine learning eli koneoppiminen on sitä, että algoritmi laitetaan etsimään kerätystä datasta korrelaatioita ja muodostamaan niistä malleja. Tätä opittua mallia voidaan jatkossa hyödyntää esimerkiksi ennusteiden laatimisessa tai kuvan tunnistuksessa tms. riippuen siitä, mitä syötedata on ollut. Jos syötteet ovat olleet kuvia ja niihin liittyviä metatietoja, kuten ’kissa’ asiasanana niille kuville, joissa esiintyy kissa, niin jatkossa mallilta voidaan kysyä, miten todennäköisesti jossain uudessa kuvassa on kissa.

Vuonna 1950 Alan Turing esitti ensimmäistä kertaa kysymyksen, että voivatko koneet ajatella. Tätä pidetään artificial intelligencen eli tekoälyn historian aloituksena. Turing loi myös testin, jolla voidaan mitata koneen suorituskykyä älykkyyden suhteen. (Mohammed & Khan & Bashier. 2016) Vaikka koneoppiminen ja tekoäly kuulostavat lähes samoilta asioilta, eivät ne kuitenkaan tarkoita samaa. Kaikki koneoppiminen on tekoälyä, mutta kaikki tekoäly ei ole koneoppimista. Tekoäly voi olla monta if-lausetta peräkkäin ja oikeastaan tekoäly on sitä, kun kone tekee jotain älykästä. (Amatriain X.) Se ei kuitenkaan tarkoita, että tekoäly oppisi mitään. Tekoälyn ja koneoppimisen ero onkin juuri siinä, että koneoppimisessa kone oppii ajan myötä tehokkaammaksi ja paremmaksi, mutta tekoälyssä koneelle on ohjelmoitu valmiiksi kaikki ”älykkyys”. Tekoäly suorittaa tehtävän aina valmiiksi koodatulla tavalla, eikä opi mitään.

Koneet voivat oppia esimerkiksi tietyn tehtävän useiden tuhansien toistojen kerryttämän datan, ja siitä mahdollisesti datatieteen avulla generoidun taulukon, graafin tai ennustuksen avulla tekemään tehtävän nopeammin, paremmin tai tehokkaammin.

Koneoppimista hyödyntäviä ratkaisuja otetaan yhä enemmän ja enemmän käyttöön käyttäjäkokemuksen, tuottavuuden ja kilpailullisen etumatkan parantamiseksi. On odotettavissa, että tulevina vuosina monilla aloilla koneoppimista käyttävien sovellusten käyttö kasvaa räjähdysmäisesti. Vuonna 2022 koneoppimisen markkina-arvo tulee olemaan jo 8,81 miljardia Yhdysvaltain dollaria, kun vuonna 2016 se oli 1,03 miljardia dollaria. (Machine Learning Market, 2017.)

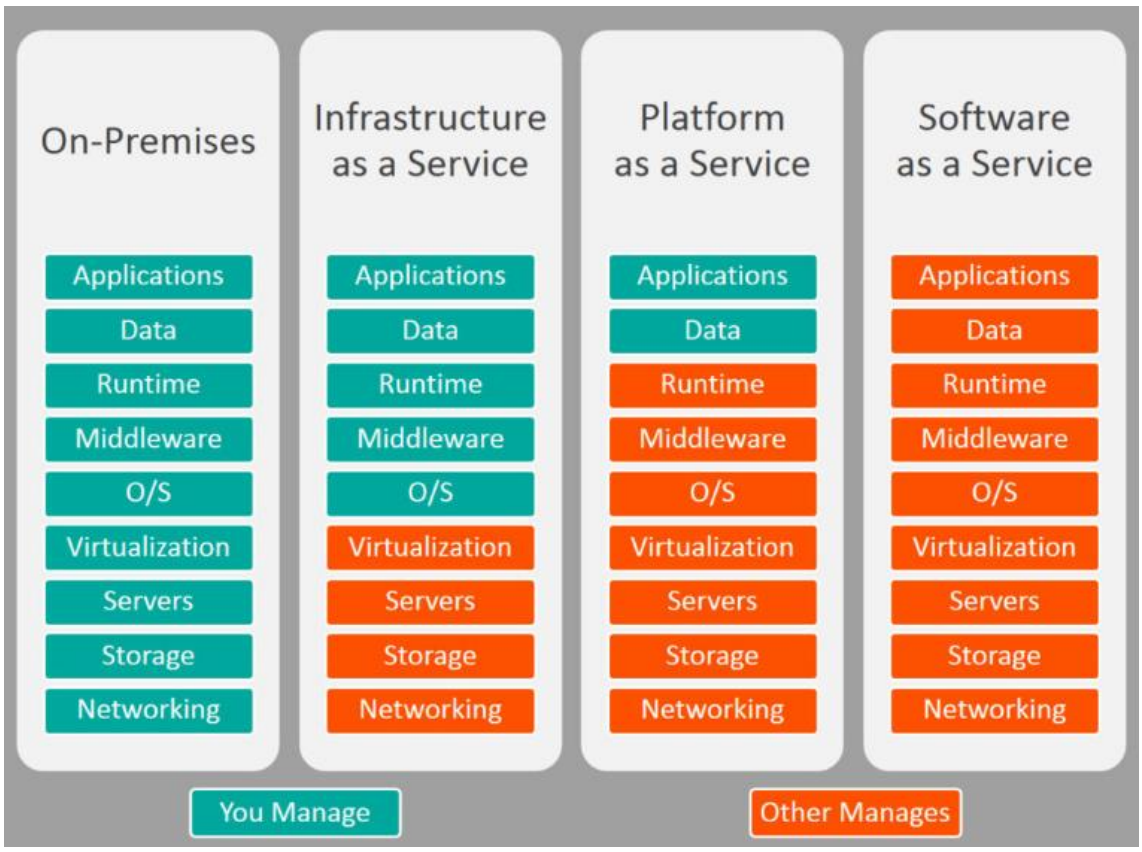
Peleissä on jo pitkään käytetty tekoälyä tietokoneen ohjaamissa boteissa, jotka pelaavat pelaajaa vastaan tai hänen kanssaan. Koneoppiminen on myös tullut mukaan ja esimerkiksi kaikkien tuntemaan Super Marioon on tehty koneoppimista käyttävä koodi. Koodin on luonut SethBling -niminen henkilö, joka on jakanut lähdekoodin avoimesti kaikkien käyttöön ja MarI/O-nimellä kulkeva koodi on myös ollut twitch.tv-nimisessä suoratoistopalvelussa ajossa. MarI/O:ssa on käytetty Neuro Evolution of Augmenting Topologies -algoritmia (KUVA 1). Käytännössä kone siis aloittaa Super Marion pelaamisen nollapisteestä: se ei tiedä mitä mikäkin painike tekee ja toistoja tarvitaan useita tuhansia, joiden aikana MarI/O oppii pelaamaan tason läpi. MarI/O tietää oppimatta, kuinka pitkälle ja kuinka nopeasti se on päässyt tiettyyn kohtaan tasoa. Näiden avulla lasketaan Fitness-arvo, jonka isoin arvo kertoo tähän mennessä parhaimman opitun tavan päästä nopeimmin pisimmälle tasossa. (MarI/O – Machine Learning for Video Games 2015.)



KUVA 1. MarI/O lukuisten toistojensa parissa. (MarI/O – Machine Learning for Video Games 2015)

2.4 Saas, paas, iaas ja niiden kehitys

SaaS eli software as a service, PaaS eli platform as a service ja IaaS eli infrastructure as a service ovat pilvipalveluiden eri malleja. Niillä jokaisella on omat hyötynsä ja käyttötarkoituksen mukaan pitää aina miettiä, mikä niistä sopisi parhaiten. Voi jopa olla ettei mikään niistä sovi kunnolla ja tehokkain ratkaisu on asentaa ja konfiguroida kaikki itse. Lyhykäisyydessään pilvipalvelumallien eroavaisuudet ovat seuraavat (KUVA 2).



KUVA 2. Iaasin, paasin ja saasin eroavaisuudet (Watts, S. 2017)

Käytännössä SaaS eli sovellus palveluna tarkoittaa, että kaikki on valmiina, eikä käyttäjän tarvitse itse huolehtia mistään muusta kuin sovelluksen käytöstä. Useimmiten sovelluksen käytöstä laskutetaan joko käytön mukaan tai kuukausimaksuilla (Sinnett, W.M. 2010. 1). Ehkä tunnetuimpia SaaS palvelumallin omaavia palveluita ovat Googlen palvelut, kuten Maps ja Drive. Myös Dropbox ja GoToMeeting toimivat SaaS palvelumallilla. (Watts, S. 2017.)

PaaS eli alusta palveluna on puolestaan nimensä mukaisesti palvelualusta, josta löytyy kaikki muu paitsi sovellukset ja niiden käyttämä data. OpenShift on hyvä esimerkki alustasta palveluna. OpenShiftissä käyttäjän ei esimerkiksi tarvitse kuin syöttää sovelluksen lähdekoodin url ja kaikki muu hoituu automaattisesti OpenShiftin puolesta. Muita esimerkkejä PaaS palvelumallista ovat Windows Azure ja Heroku. (Watts, S. 2017.)

IaaS eli infrastruktuuri palveluna luopuu muutamasta palasesta, jotka PaaS tarjoaa. Palvelimet, tallennustilat, virtualisaatio ja verkkokytkennät hoituu palveluntarjoajan puolesta ja muut, jopa käyttöjärjestelmät pitää hoitaa itse. Täten IaaS onkin palvelumalleista

joustavin ja skaalautuvin. Hyviä esimerkkejä IaaS palvelumallista ovat DigitalOcean ja Amazon Web Services. (Watts, S. 2017.)

2.5 Openshift, kontit, Platform as a service

OpenShift on Red Hatin vuonna 2011 julkaisema sovellusten ajamiseen ja hallintaan tarkoitettu PaaS. OpenShift toimii Dockerin konttien ja Kuberneteksen yhteisessä harmoniassa, missä Dockerin kontit toimivat sovellusten ajamisalustana ja Kubernetes aikatauluttaa ja ohjaa kontteja sekä niiden toimintaa. OpenShiftistä on olemassa useita eri versioita: OpenShift Origin, RedHat OpenShift Online, OpenShift Dedicated, OpenShift Container Platform ja OpenShift.io. Näistä OpenShift Onlinea pystyy käyttämään ilmaiseksi tietyin rajoituksin. (OpenShift Products.) Opinnäytetyössä tullaan kuitenkin perehtymään OpenShift Container Platformiin, joka on toimeksiantajalla laajassa käytössä.

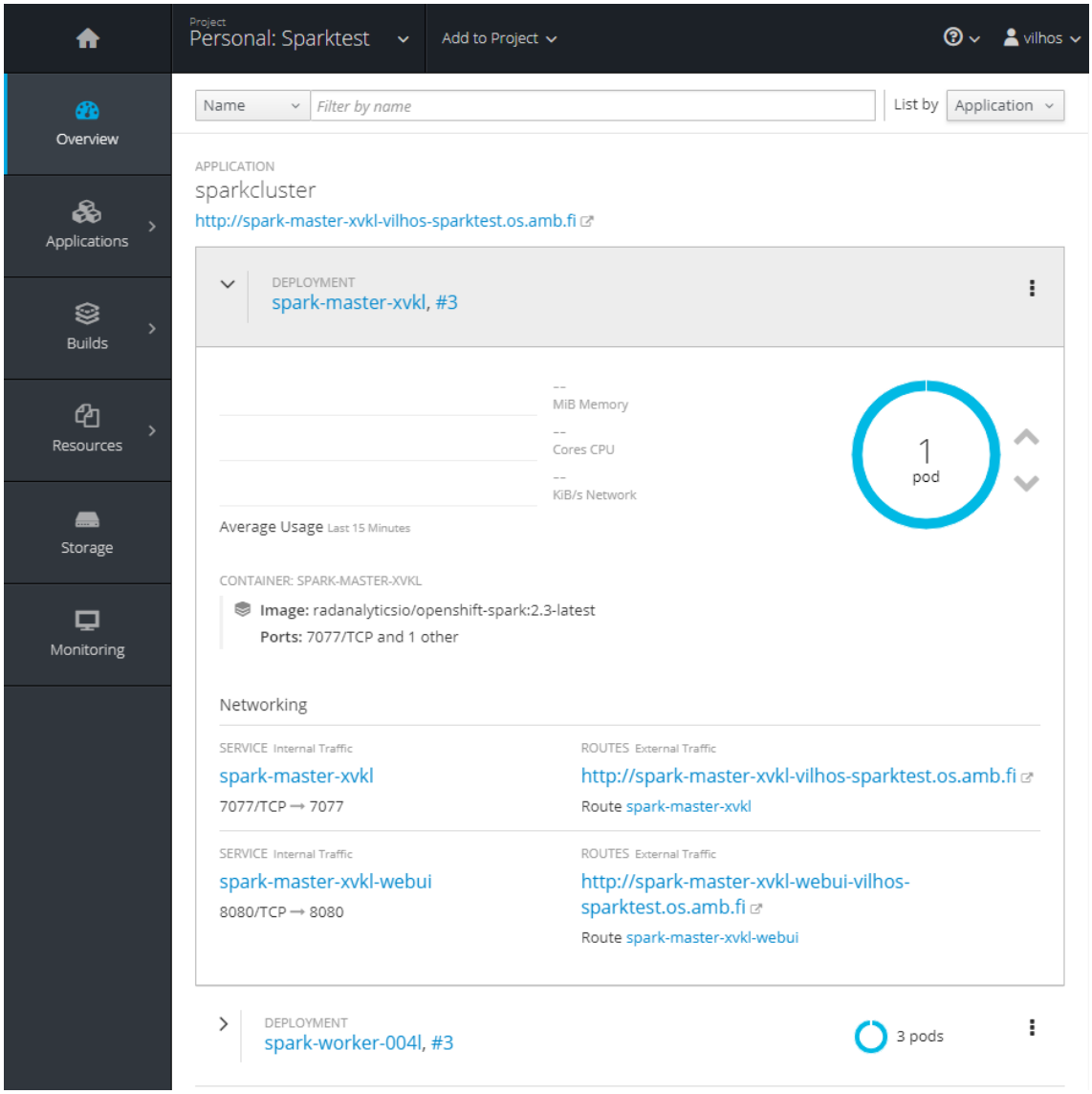
Docker containerit eli kontit ovat avainasemassa OpenShiftissä. Kaikki OpenShiftiin ajoon laitettavat sovellukset ovat Dockerin imageista rakennettuja kontteja. Docker image koostuu useista eri layereistä eli tasoista (KUVA 3). OpenShiftissä on käytössä runsaasti s2i eli source to image -templateja monelle eri ohjelmointikielelle. S2i templaten avulla pelkällä lähdekoodilla saadaan luotua image, josta taas rakennetaan kontti, joka laitetaan OpenShiftiin ajoon. Konttien ideana on, että kontin voi viedä kehitysympäristöstä suoraan tuotantoympäristöön ja kontissa ajossa olevat sovellukset toimivat täysin samalla tavalla ympäristöstä huolimatta (Raj, P. ym. 2016.)

```
1  Cloning "ssh://git@bitbucket.ambientia.fi:7999/~vilhos/react-leaflet.git" ...
2          Commit: 3ff2od8n56hxreqwyr (Modify htaccess file)
3          Author: Vilho Stenman <vilho.stenman@ambientia.fi>
4          Date:   Fri Sep 21 15:28:18 2018 +0300
5  ---> Enabling s2i support in httpd24 image
6  AllowOverride All
7  ---> Installing application source
8  => sourcing 20-copy-config.sh ...
9  => sourcing 40-ssl-certs.sh ...
10 Pushing image 192.54.54.22/vilhos-leaflet/apache-react:latest ...
11 Pushed 4/5 layers, 81% complete
12 Pushed 5/5 layers, 100% complete
13 Push successful
```

KUVA 3. Docker image build ja push OpenShiftissä.

OpenShiftiä on mahdollista käyttää, joko web-käyttöliittymän avulla (KUVA 4), tai komentorivityökalulla oc. Komentorivityökalun voi asentaa Linuxille, Macille ja Windowsille. Red Hatin kautta komentorivityökalun lataaminen vaatii Red Hat käyttäjäkirjautumisen ja aktiivisen OpenShift Enterprise tilauksen. (Get Started with the CLI).

Käytännössä OpenShiftiä voi käyttää kumman tahansa työkalun avulla, mutta komentorivityökalulla voi tehdä enemmän kuin web-käyttöliittymällä. Esimerkiksi tietyn deploymentin kokonainen poistaminen hoituu yhdellä komennolla, ja Docker hubista docker-imageiden OpenShiftiin ajaminen onnistuu helposti yhdellä komennolla. Toki on myös täysin käyttäjäkohtaista kumpaa käyttää. Web-käyttöliittymä kuitenkin mahdollistaa esimerkiksi koko projektin kaikkien deploymenttien yhdenaikaisen tilanseurannan ja podien logien reaaliaikaisen seurannan.



KUVA 4. OpenShift Container Platform web-käyttöliittymä.

OpenShift on sovelluskehittäjälle todella hyvä alusta sovellusten kehityksen, testaamisen ja lopullisen tuotantoajon kannalta. OpenShift ja dockerin kontit mahdollistavat helposti yhdelle projektille monen samanaikaisen sovelluksen ajamisen. OpenShiftissä voi suoraa esimerkiksi antaa templatelle, eli sovelluspohjalle lähdekoodin versionhallinta-url:n ja template luo sovellukselle ympäristön, jossa sovellus käännetään ja ajetaan omassa podissaan. Templaten kautta on myös mahdollista luoda sovelluksen rinnalle esimerkiksi tietokanta ajoon omaan podiinsa ja luoda sovellukselle julkinen www-osoite, josta sovellukseen pääsee käsiksi. Sovellukselle voi myös luoda hookin eli ”koukun”, jonka avulla sovellus käännetään aina uudelleen kun versionhallinnassa tapahtuu muutoksia. Täten sovelluksen ajaminen OpenShiftissä on myös todella helppoa ja mutkatonta jo sovelluksen kehitysvaiheessa, eikä pelkästään tuotantokäytössä. OpenShift mahdollistaa vaikka mitä muutakin, kuten konfiguraatiokarttoja eli ConfigMaps, sovellusten ”terveystarkatuksia” eli Application health checks ja ajoitettuja komentoajoja eli Cron Jobs.

2.6 Koneoppimisen ja datatieteen hyödyt Openshiftissä

Koneoppimisen ja datatieteen ratkaisuilla ei etsitä hyötyjä itse OpenShiftin ympäristöön ja toimintatapoihin, vaan ennemminkin, mitä siitä hyödytään, että ne ovat juuri OpenShiftissä ajossa.

Isoin hyöty OpenShiftistä koneoppimisen ja datatieteen ratkaisuille on, että OpenShiftin ympäristössä on todella helppo luoda yhdelle projektille monta eri koneoppimisen ja datatieteen sovelluksen asennusta, jolloin sovellusten välinen kommunikointi ja datan siirrot onnistuvat hyvin helposti esimerkiksi paikallisten IP-osoitteiden avulla. OpenShiftiin on myös mahdollista luoda templateita, jotka mahdollistavat sen, että yksi template luo kerrallaan kokonaisen koneoppimis ja datatiede -putken. Templatelle ei välttämättä tarvitse antaa kuin datan lähde parametrina ja template hoitaa käyttäjän puolesta kaikki muut konfiguraatiot ja manuaaliset asennustyöt.

OpenShiftin web-käyttöliittymän kautta on helppo seurata asennusten tilaa ja konfiguroida olemassa olevia asennuksia. On myös se etu, että omassa yksityisessä pilvessä eli tässä tapauksessa OpenShiftissä käyttäjällä on kaikki oikeudet ja mahdollisuudet muokata järjestelmää ja asennuksia sellaisilla tasoilla, mitä kolmannen osapuolen kuten Amazon Web Servicen alustaa käytettäessä ei olisi mahdollista.

3 KONEOPPIMISEN JA DATATIETEEN RATKAISUT

3.1 Johdanto

Opinnäytetyön tavoitteen mukaisesti tässä luvussa selvitetään, miten saada koneoppimisen ja datatieteen toteutuksia ajettua OpenShiftin päällä (sovelluskehittäjän näkökulmasta) mahdollisimman helposti. Käytännössä siis monien eri koneoppimisen ja datatieteen sovellusten ja rajapintojen asennusten testaamista ja konfigurointia, niin että sovellus tai rajapinta toimii OpenShiftissä. Tässä luvussa käydään läpi näiden sovellusten ja rajapintojen asennukset yksityiskohtaisesti, jotta myöhemmin näiden sovellusten tai rajapintojen asennuksiin voisi käyttää suoraan näitä ohjeita.

Muutamat asennukset tehdään suoraan radanalytics-yhteisön tuottamien templateiden ja docker-imageiden pohjalta. Radanalytics on Red Hatin luoma yhteisö, jonka tarkoituksena on kehittää älykkäiden sovellusten ajoa OpenShiftin päällä. Yhteisö on tuottanut useita ohjeita ja projekteja koneoppimisen ja datatieteen ratkaisuiden OpenShiftiin asennuksen ja integroinnin helpottamiseksi. Kaikki radanalyticsin projektit ja niiden tuotokset ovat avointa lähdekoodia ja vapaasti käytettävissä. (Radanalytics.)

3.2 Datatiede ja koneoppimis -ketjun kuvaus

Datatiede ja koneoppiminen vaativat dataa toimiakseen. Datatiede- ja koneoppimisketjusta itse datatieteen ja koneoppimisen osuudet ovat vain yksi osa ja niitä ennen datan saamiseen ja prosessointiin vaaditaan suuri määrä työtä. Aivan aluksi tarvitaan datan lähde, joka on mahdollisesti suuri määrä eri antureita, jotka keräävät esimerkiksi tuulivoimalan tietoja. Antureilla kerättävää dataa pitää putkittaa monen eri prosessin läpi ennenkö sitä voi oikeasti käyttää koneoppimiseen. Datan lähde voi myös olla jo valmiiksi kerätty aineisto, joka ajetaan kertaluontoisesti putken läpi.

On olemassa ”OSEMN pipeline” -käsite, jossa jokainen kirjain tarkoittaa omaa vaihettaan putkessa. O eli obtain tarkoittaa datan keräystä. S eli scrub tarkoittaa datan puhdistusta, joka käytännössä tarkoittaa turhien tai virheellisten sarakkeiden ja arvojen poistamista tai muokaamista ja puuttuvien arvojen täydentämistä. Seuraavassa vaiheessa E eli exploring

päästäänkin jo datatieteen osuuteen, vaikka datatieteen osuus voidaan ajatella alkavan jo kohdasta S. Dataa visualisoidaan ja datasta etsitään tietynlaisia poikkeamia, sarjoja tai jatkuvuuksia. Näitä käyttämällä saadaan koneoppiminen mukaan, kun saavutaan putken kohtaan M eli modeling. Esimerkiksi poikkeamia käyttämällä voidaan ennustaa seuraavaa poikkeamaa ja näin ollen esimerkiksi ennaltaehkäistä tulevaa ongelmaa. Viimeinen vaihe N eli interpreting on käytännössä saatujen tulosten esittelyä tavalla tai toisella. (Lao, R. 2018.)

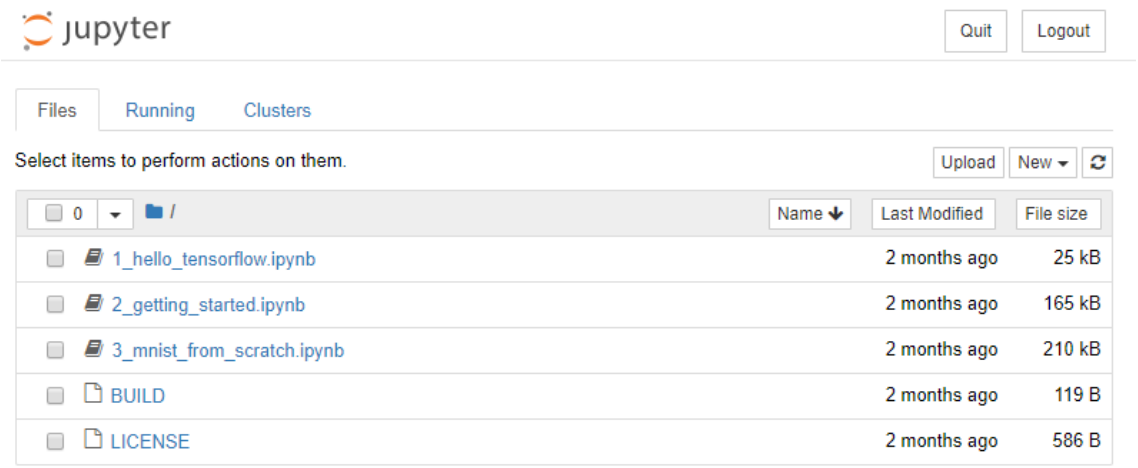
3.3 TensorFlow

TensorFlow on Googlen kehittämä sovellusrajapinta koneoppimisen mallien luomiseen ja asentamiseen. TensorFlow oli alunperin kehitetty Googlen sisäiseen käyttöön koneoppimis-applikaatioiden luomista varten. Vuoden 2015 lopulla Googlella kuitenkin päätettiin julkaista TensorFlow muun maailman käyttöön avoimen lähdekoodin applikaationa. TensorFlow on lisensoitu Apache open source software lisenssin alle ja sen kaikenlainen muokkaus, käyttö ja jopa uudelleen jakaminen on sallittua, ilman minkäänlaisia kuluja. TensorFlow on suunniteltu toimimaan suurien datamäärien kanssa. Kaikki TensorFlown kanssa käytettävä data tulee olla moniulotteisessa taulukossa, jota kutsutaan myös nimellä tensor. Operaatioita suoritettaessa tulee käyttää vuokaavion eli flow chartin tapaista laskennallista graafia. TensorFlowin nimi tuleekin siis näppärästi järjestelmän tuotesuunnittelun pohjalta. (Hope, T. & Resheff, Y, S. & Lieder, I. 2016.)

3.4 TensorFlowin asennus ja käyttöönotto

TensorFlowin asennusta ei saatu toimimaan toimeksiantajan OpenShift ympäristössä, joten asennus tehtiin lokaalissa OpenShift ympäristössä nimeltään MiniShift. MiniShiftin saa asennettua MiniShift CDK -asennustyökalun avulla, jonka saa ladattua Red Hatin sivuilta kirjautumalla Red Hat Developer Program -tunnuksilla sisään. Minishiftin asennus on melko yksityiskohtainen ja asennus vaatii Hyper-V virtuaalikytkimen konfiguroinnin, ympäristömuuttujien asettamista ja n.20 gigatavua levytilaa. MiniShiftin asennusohjeet löytyvät työkalun lataussivulta. Opinnäytetyössä käytetty MiniShift asennettiin Windows 10 -käyttöjärjestelmälle.

MiniShiftin asennuksen jälkeen web-käyttöliittymä ja oc-komento toimivat täysin samoin kuin toimeksiantajan OpenShift ympäristössä. Projektin luonnin jälkeen tarvitsee sallia root-käyttäjän käyttö projektilla komennolla *oc adm policy add-scc-to-user anyuid -z default*, jonka jälkeen TensorFlowin voi asentaa komennolla *oc new-app tensorflow/tensorflow*. TensorFlowin asennuksen jälkeen tulee vielä avata portti 8888 liikenteelle komennolla *oc expose dc tensorflow --port=8888 --name=jupyter*. Asennuksen ja portin avaamisen jälkeen TensorFlowin mukana tulevaan Jupyter notebook web-sovellukseen (KUVA 5) pääsee OpenShiftin käyttöliittymässä näkyvän url:n kautta käsiksi.

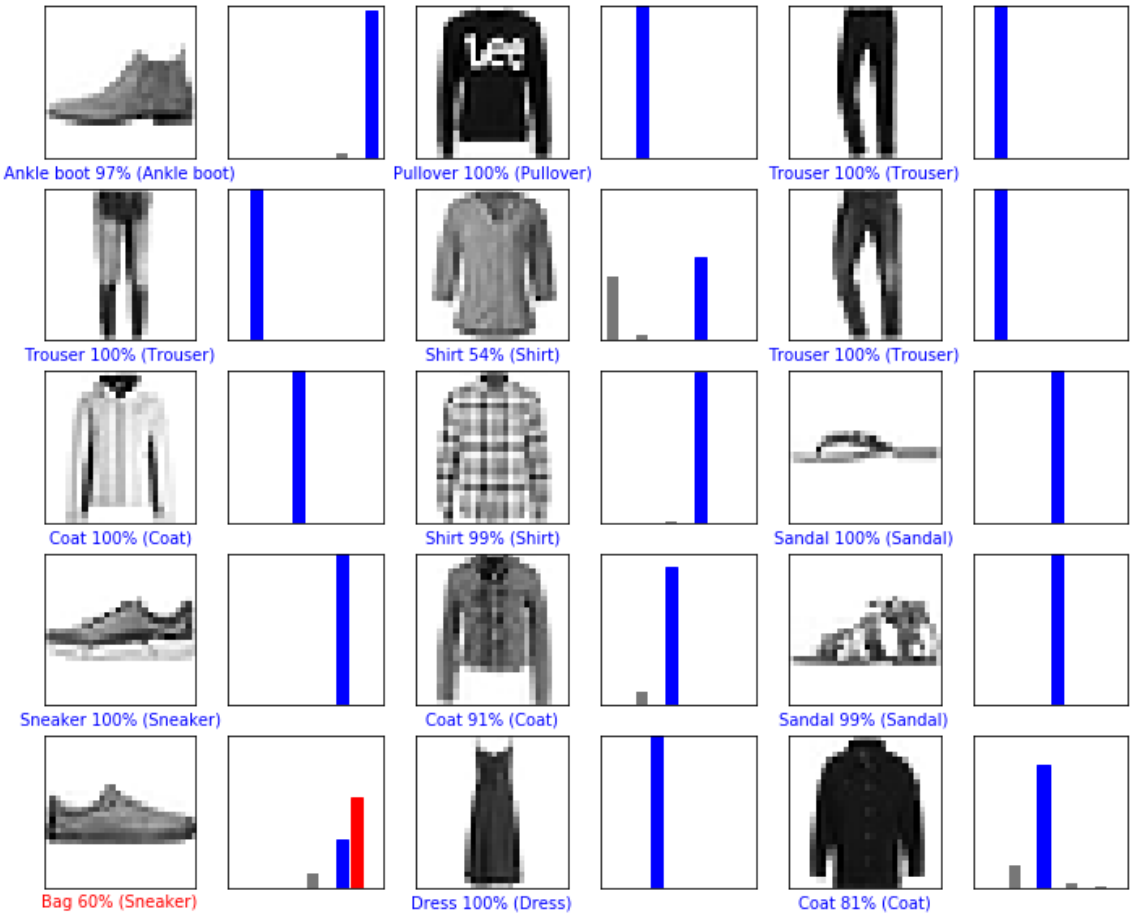


KUVA 5. Jupyter Notebook web-käyttöliittymä.

TensorFlowilla ja Jupyter Notebookin avulla voidaan tehdä monenlaisia ennustuksia, malleja ja kaavioita koneoppimista ja datatiedettä hyväksikäyttäen. TensorFlowin virallisilta sivuilta löytyvistä tutoriaaleista ajoin omassa ympäristössäni esimerkin ”Train your first neural network: basic classification”.

Tutoriaalissa käytetään lähdedatana 70 000 kappaletta mustavalkokuvia vaatteista, jotka koostuvat 10 eri kategoriasta. Näistä kuvista 60 000 kappaletta käytetään neuroverkon opettamiseen ja loppuja 10 000 kappaletta siihen, kuinka hyvin neuroverkko oppi tunnistamaan kuvia. Jotta neuroverkolle voidaan opettaa kuvien tunnistamista, täytyy ensin esikäsitellä data, konfiguroida neuroverkon malli ja koota malli. Opetuksen jälkeen testataan, kuinka hyvin neuroverkon malli osaa tunnistaa 10 000 kuvasta jokaiselle kuvalle oikean kategorian. Malli pääsee opetuksen jälkeen systemaattisesti noin 88 prosentin tarkkuuteen, joka tarkoittaa sitä, että suurin osa kuvista menee oikein ja pieni osa väärin. Kuitenkin vaikka neuroverkko tunnistaisi kuvan oikein, voi se silti olla melko epävarma

veikkaus. Esimerkiksi yksi kuva voi mennä 54% varmuudella oikein ja toinen 100% varmuudella. Neuroverkko voi siis kertoa, että tietty kuva on 54% varmuudella paita, 40% varmuudella t-paita ja 6% varmuudella housut. (Train your first neural network: basic classification.) Sininen väri kuvaa oikein ennustetun kategorian määrää ja punainen väärin ennustetun kategorian määrää (KUVA 6).



KUVA 6. Opetetun neuroverkon mallin ennustukset yksittäisten kuvien osalta.

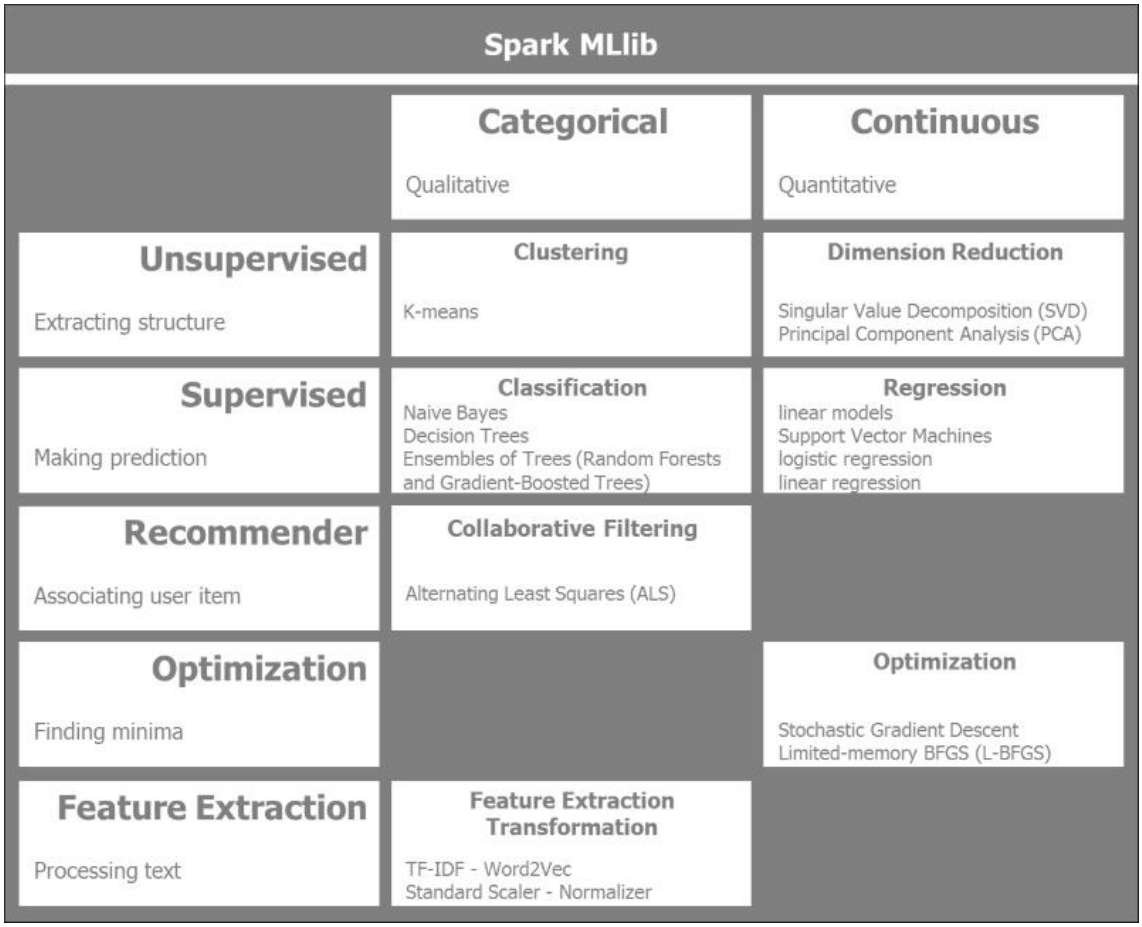
3.5 Apache Spark

Apache Spark on laskentaan pohjautuva rajapinta big datan nopeaa prosessointia varten ja se on suurin avoimen lähdekoodin projekti big dataan liittyen. Sparkin kehitys aloitettiin vuonna 2009 Kalifornian yliopiston AMPLabissa ja jo seuraavana vuonna siitä tuli täysin avointa lähdekoodia, kun se lahjoitettiin Apachelle. Sparkkia kehitetään yhä jatkuvasti ja siitä julkaistaan uusi versio aina kolmen kuukauden sykleissä. (Liu. 2016.)

Sparkin luvataan olevan, ja on testattu olevan yli 100 kertaa nopeampi, kuin Hadoop MapReduce ja sillä on monia eri paremmuuksia MapReduceen ja muihin dig data -laskentarajapintoihin nähden. Spark koostuu Sparkin ”core enginestä” ja neljästä ohjelmakirjastosta, mitkä muodostavat keskenään yhteisen alustan. Neljä ohjelmakirjastoa ovat Spark SQL, Spark Streaming, Mlib ja GraphX ja niillä kaikilla on Python, Java ja Scala -ohjelmointikielien APIt. (Liu. 2016.)

Sparkin laskennassa hyödynnetään muistiin hajautettua laskennallista lähestymistapaa, joka tekee Sparkin laskennasta yhden nopeimmista, etenkin iteratiiviseen laskentaan (Liu. 2016).

Sparkissa oleva MLib on koneoppimisen ohjelmakirjasto, missä on lukuisia koneoppimisen algoritmeja, joilla voidaan suorittaa seuraavia tapahtumia (KUVA 7).



KUVA 7. MLibin sisäänrakennetut algoritmit. (Nandi. 2015)

Sparkia voisi käyttää melko erikoisissakin käyttötarkoituksissa. Radek Ostrowski kertoo artikelissaan kuinka Sparkin Mlib koneoppimis-ohjelmakirjaston avulla voisi analysoida

Twitterin tweettejä ja poimia joukosta sellaiset tweetit, jotka liittyvät maanjäristyksiin. Poimitut tweetit analysoitaisiin sekä opetettaisiin mallille ja lopputuloksena voitaisiin tweettien perusteella saada informaatio maanjäristyksestä nopeammin kuin virallisilta tahoilta. (Ostrowski, R.)

3.6 Apache Sparkin asennus ja käyttöönotto

Apache Sparkin asennuksen OpenShiftiin voi suorittaa monilla tavoilla. Opinnäytetyötä tehdessäni kävin läpi kaksi vaihtoehtoa, joista molemmat toimivat ilman lisäkonfiguraatioita. Apache Spark osoittautui olemaan ainut ratkaisu, joka toimi toimeksiantajan OpenShift ympäristössä.

Ensimmäisessä asennuksessa käytin radanalyticsin OpenShift Spark dockerfileä, joka löytyy Docker Hubista. OpenShiftin komentorivityökalulla Sparkin sai asennettua yhdellä komennolla *oc new-app radanalytics/openshift-spark* (KUVA 8).

```
C:\Users\vilhos\Documents\openshift-ja-docker-setit>oc new-app radanalyticsio/openshift-spark
--> Found Docker image 10b9516 (5 weeks old) from Docker Hub for "radanalyticsio/openshift-spark"

    * An image stream will be created as "openshift-spark:latest" that will track this image
    * This image will be deployed in deployment config "openshift-spark"
    * The image does not expose any ports - if you want to load balance or send traffic to this component
      you will need to create a service with 'expose dc/openshift-spark --port=[port]' later

--> Creating resources ...
    imagestream "openshift-spark" created
    deploymentconfig "openshift-spark" created
--> Success
    Run 'oc status' to view your app.
```

KUVA 8. Radanalyticsin OpenShift Spark imagen asennus.

Asennus onnistuu ja Spark toimii OpenShiftissä kuten on tarkoitus, mutta dockerfilen kautta luodessa Sparkille ei generoidu serviceä, routeja eikä workereitä, vaan pelkkä spark-master pod. Tästä syystä seuraavaksi esiteltävä templaten avulla tapahtuva Sparkin asennus on monilta osilta toimivampi ja vähemmän lisäkonfiguraatiota vaativa.

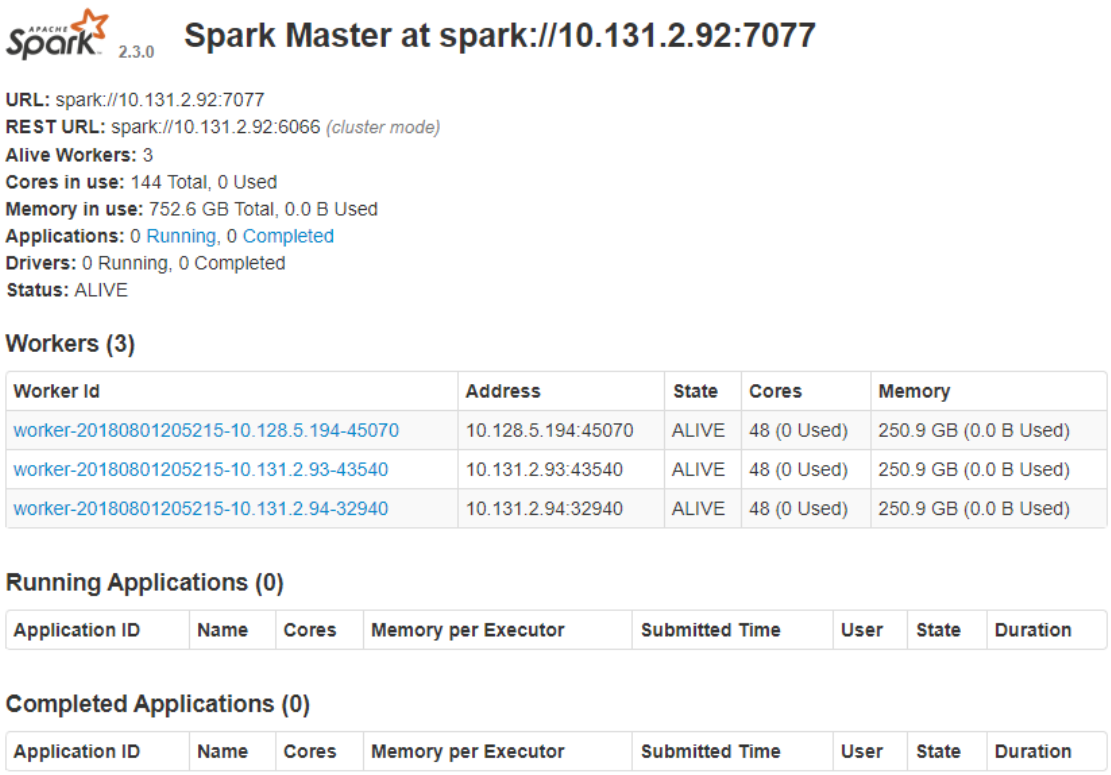
Seuraavana asensin saman radanalyticsin OpenShift Sparkin, mutta tällä kertaa käyttäen projektin GitHub sivuilta löytyvää templatea, jonka mukana Sparkiin tulee workereitä, servicet ja routet, joista pääsee Sparkin web-käyttöliittymään käsiksi. Kun luo OpenShiftiin applikaatioita templaten kautta, saa käyttäjä itse päättää tiettyjä arvoja ja saattaa joutua syöttämään joitakin tietoja ennen asennusta. Useimmat kentät voi silti jättää tyhjiksi, sillä niihin generoituu arvo itsestään, jos ne jättää tyhjiksi. Templaten kautta asennusta

tehtäessä, asennus hoituu mielestäni helpoiten web-käyttöliittymän kautta, johon template pitää toki ensin itse tuoda. Templaten voi tuoda, joko tietyn projektin nimiavaruuteen tai vaikka koko OpenShift asennuksen nimiavaruuteen, jolloin se näkyy kaikille samaa OpenShiftiä käyttäville.

Templaten tuominen tiettyyn projektiin onnistuu komentoriviltä ja web-käyttöliittymästä. Komentoriviltä se onnistuu komennolla *oc create -f <filename>*, jossa filename on templaten yaml- tai json-tiedoston nimi ja pääte. Web-käyttöliittymässä tiedoston voi yksinkertaisesti valita tai vaikka raahata ja tiputtaa sille varattuun kenttään. Templaten tuomisen jälkeen web-käyttöliittymästä siihen pääsee käsiksi avaamalla tietyn projektin, jonka jälkeen yläreunasta *Add to Project > Browse Catalog*. Hakukenttään kirjoittamalla template tulee näkyviin.

OpenShift Sparkin template vaatii Sparkin master/worker imagen nimeä, joka on valmiiksi täytettynä *radanalyticsio/openshift-spark:2.3-latest*. Template kysyy myös master servicen ja workerin nimiä, jotka generoituvat itsestään, ellei nimiä halua itse antaa. Lopuksi templatessa näytetään myös luodut labelit ja käyttäjä voi halutessaan muokata niitä ja lisätä uusia. Create nappia painettua projektille syntyy Sparkin master service, jota ajetaan yhdellä podilla ja kolme workeria, joille jokaiselle on oma podinsa.

Jotta Sparkiin pääsee käsiksi web-käyttölittymän kautta, tulee luoda route kohdasta *Applications > Routes > Create route.* Routelle tulee antaa nimi, url-osoite ja service johon route ohjaa. Nimeksi voi antaa oikeastaan mitä vaan esimerkiksi *spark-master-xvkl-webui*, mutta service listasta tulee valita juurikin *webui*-päätteinen service. Kun routen on saanut luotua tulisi siihen navigoimalla saada näkyviin seuraavanlainen näkymä (KUVA 9).



KUVA 9. Apache Sparkin web-käyttöliittymä.

Käyttöliittymästä näkee workereiden id:n, osoitteen, tilan, käytössä olevien ytimien määrän ja käytössä olevan muistin määrän. Workereiden id:tä klikkaamalla avautuu worker-kohtainen näkymä, josta saa selville onko workerillä tällä hetkellä jotain suoritettavana.

Testasin Sparkin toiminnallisuutta OpenShiftin web-ui:n kautta avaamalla spark-master podin terminaalin. Terminaalissa annoin ensin komennon *cd $SPARK_HOME*, jolla pääsee Sparkin asennuskansioon. Seuraavaksi ajoin esimerkkikansiosta koodin, jolla Spark arvioi piin arvon ”heittämällä tikkoja” neliön sisällä olevaan ympyrään. Neliön ja ympyrän sisälle osuneiden tikkojen määrillä saadaan arvioitua piin arvo. (Apache Spark Examples.) Koodin esimerkkikansioista voi ajaa komennolla *./bin/spark-submit examples/src/main/python/pi.py* ja muutaman kerran koodin ajettuani Spark arvioi piin olevan 3.150640, 3.130800 ja 3.141520, jotka ovat melko epätarkkoja arvoja, mutta silti suuntaa-antavia. Itse luodun koodin, joka on Scala, Python tai Java -ohjelmointikieltä voi myös tuoda omaan kansioonsa ja ajaa koodin samalla tavalla kuin edeltävän esimerkin.

Sparkia voi myös käyttää avaamalla podin terminaalin ja antamalla komennon *pyspark* (KUVA 10). Sparkin käynnistyttyä voi koodia kirjoittaa python-kielellä rivi kerrallaan.

Sparkin voi myös käynnistää komennolla *./bin/spark-shell*, jolloin Sparkista käynnistyy Scala-ohjelmointikielen versio.

```
sh-4.2$ pyspark
Python 2.7.5 (default, Jul 13 2018, 13:06:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
18/11/03 21:39:51 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
ava classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 2.7.5 (default, Jul 13 2018 13:06:57)
SparkSession available as 'spark'.
>>>
```

KUVA 10. Sparkin Python versio käynnistettynä terminaalissa.

3.7 Mahdolliset muut ratkaisut ja niiden läpikäynti

Alkueräisen tavoitteen mukaisesti tarkoituksena oli saada datatiede- ja koneoppimisratkaisuita asennettua Ambientian OpenShift ympäristöön. Lähes kaikki ratkaisut vaativat kuitenkin root-oikeudet toimiakseen, joita ei Ambientian OpenShift ympäristössä, kuten ei myöskään misään muussakaan OpenShift ympäristössä lähtökohtaisesti ole konteissa ajettaville sovelluksille myönnetty.

Root-oikeudet pystyy kuitenkin antamaan esimerkiksi yhdelle projektille kerrallaan, mutta Ambientian käytäntönä on, että root-oikeuksia ei myönnetä projekteille turvallisuussyistä. Myös dockerfileä eli sovelluksen docker imagen rakentamiseen käytettävää konfiguraatiotiedostoa muokkaamalla on mahdollista saada sovellus toimimaan ilman, että se käyttää root-käyttäjää. Opinnäytetyötä tehdessäni yritin saada useita ratkaisuita dockerfileä muokkaamalla ja useita eri versioita käyttämällä toimimaan Ambientian OpenShift ympäristössä tuloksetta.

4 RATKAISUIDEN AJAMINEN OPENSHIFTISSÄ

Tässä luvussa pohditaan ja otetaan selvää asennettujen ratkaisuiden eroista. Alunperin oli myös tarkoituksena ottaa selvää ja kokeilla, miten ratkaisuita saisi optimoitua OpenShiftissä, niin että niiden käyttö tai tehokkuus jollain tavalla parantuisi. Optimointiin ei jäänyt juurikaan aikaa, koska se olisi vaatinut laajaa testaamista ja monien asioiden tarkkaa monitorointia. Optimointeja oli alun perin tarkoitus suorittaa ja toteuttaa eri asennustapojen välillä esimerkiksi dockerfilejä muokkaamalla.

4.1 Ratkaisuiden vertailu

Apache Spark ja TensorFlow soveltuvat molemmat datatieteeseen ja koneoppimiseen. Molemmat alustat tukevat Pythonia sovellusten ohjelmointiin, mikä on hyvä asia Pythonin ollessa yleisin ja eniten käytetty ohjelmointikieli kaikkeen datatieteeseen, AI:seen ja koneoppimiseen liittyvässä ohjelmoinnissa. Pythonille on todella kattava tarjonta kirjastoja ja rajapintoja koneoppimista ja tekoälyä varten, sekä se on myös hyvin yksinkertainen ohjelmointikieli, jota kokemattomankin ohjelmoijan on helppo ymmärtää. (Protasiewicz, 2018).

Yleistasolla TensorFlow on mukavampi käytettävyydeltään, sillä itse sovelluksen käyttö onnistuu kokonaan graafisen käyttöliittymän kautta. Sparkissa puolestaan käyttöliittymän kautta näkee vain workereiden käyttöasteet ja kaikki muu sovelluksen käyttö tapahtuu terminaalin kautta komentorivikomennoin.

Vaikka molemmat ratkaisut soveltuvat sekä koneoppimiseen, että datatieteeseen on Apache Spark parempi vaihtoehto kokonaisen datatiede putkituksen käyttämiseen. TensorFlow puolestaan keskittyy enemmän koneoppimiseen ja deep learningiin, eli syväoppimiseen, jossa neuroverkkoja hyödynnetään usealla tasolla.

4.2 Yhteenveto

Todella monet ratkaisut käyttävät root-käyttäjää asennuksen aikana. Ja koska root-käyttäjänä sovellusten ajaminen podeissa ole sallittua toimeksiantajan OpenShiftissä, useita eri ratkaisuja jäi kokonaan asentamatta ja testaamatta. Näihin ratkaisuihin lukeutui mm. TensorFlow, Caffe, shogun, Pandas, Theano ja H2O. Osaan näistä oli jopa Red Hatin sivuilta löytyviä ohjeita, kuinka saada ratkaisu toimimaan OpenShiftissä, mutta useiden eri asennustapojen, dockerfile muutosten ja muiden konfiguraatioiden jälkeen ratkaisut eivät silti toimineet toimeksiantajan OpenShift ympäristössä.

Eri ratkaisuiden käyttöä olisi ollut myös todella mielenkiintoista kokeilla ja mahdollisesti myös kokeilla Tensorfowin ja Apache Sparkin yhdistelmää, joko itse yhdistelemällä molempia keskenään tai käyttämällä niistä koostettua avoimen lähdekoodin TensorFlowOnSpark ratkaisua. TensorFlowOnSpark ratkaisussa on käytetty Apache Sparkia datan prosessointiin, sekä putkittamiseen ja TensorFlowia koneoppimiseen, sekä syväoppimiseen. (TensorFlowOnSpark.)

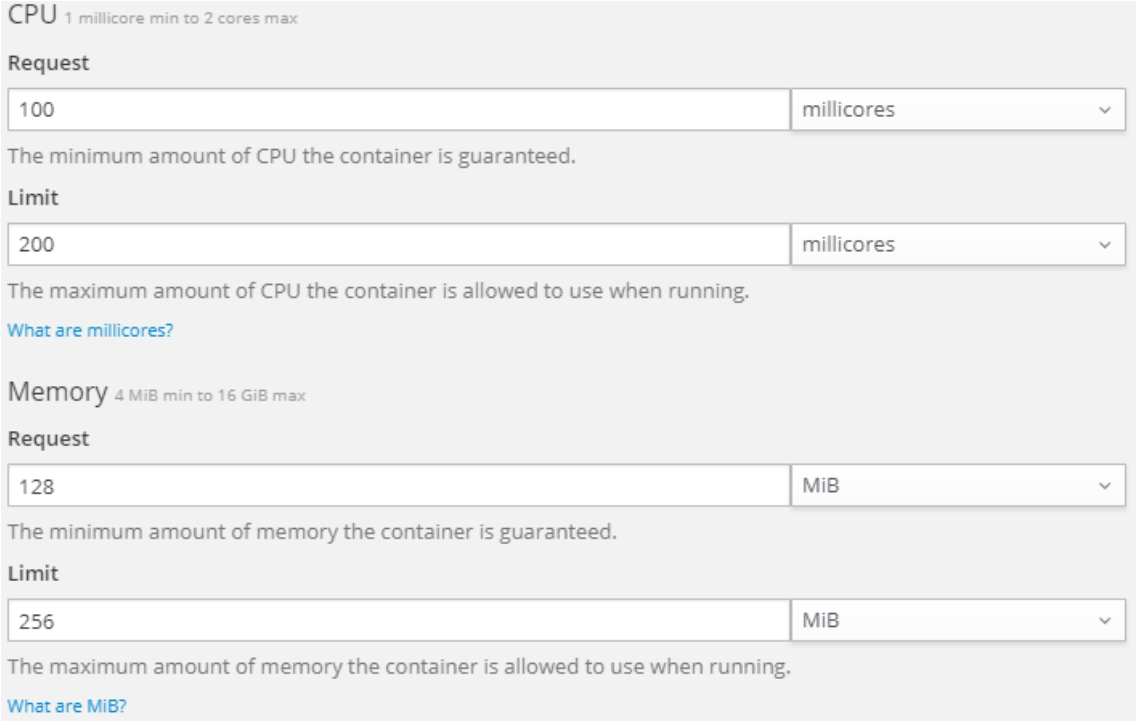
5 SKAALAUTUVUUS JA INFRASTRUKTUURIN PERUSTAMINEN

Tässä luvussa vertaillaan yksityisen pilven ja julkisen pilven erovaisuuksia skaalautuvuuden, sekä ylläpidon ja perustamisen vaivan kannalta. Luvussa myös käydään läpi, mitä skaalautuvuudella tarkoitetaan ja mitä eri skaalauskeinoja on olemassa.

5.1 Skaalautuvuus yksityisessä pilvessä verrattuna julkiseen pilveen

Skaalautuvuus on hyvin yleinen kysymys pilvipalveluiden ja etenkin cloud computingin eli pilvilaskennan parissa. Käytännössä skaalautuvuudella tarkoitetaan, että jokaista sovellusta tai infrastruktuurin palasta voidaan kasvattaa, jotta voidaan selviytyä lisääntyneestä kuormasta. Esimerkkitilanteessa melko tuntematon verkkosivusto on esillä tunnetulla verkkosivustolla, jonka vuoksi tuntemattoman verkkosivun kävijämäärä eli kuorma kasvaa ennalta-arvaamattomiin lukemiin. Jos verkkosivusto on skaalautuva, sen resurssit skaalautuvat ylöspäin ja kuormasta selvitään. Skaalautuvuutta voidaan lisätä käytännössä neljälle eri resurssille: levynopeuteen, muistin määrään, verkon nopeuteen ja prosessointikykyyn. (What is scalability in cloud computing?).

Yksityisen pilven, kuten toimeksiantajan OpenShiftin skalaalautuvuutta voi oikeastaan suorittaa kahdella eri tavalla. Yksittäisiä sovelluksia voi tiettyjen rajojen puitteissa skaalata vertikaalisesti (vertical scaling), missä nykyisen instassin resursseja kasvatetaan (What is scalability in cloud computing?). Vertikaalista skaalausta voi suorittaa manuaalisesti OpenShiftissä kasvattamalla sovelluksen resurssirajoja. Toimeksiantajan OpenShiftissä CPU:n eli prosessoriytimien maksimiraja on 2 ydintä ja muistin maksimiraja 16 gigatavua (KUVA 11).



KUVA 11. OpenShiftin resurssirajojenhallintasivu.

Toinen ja myös pitkällä tähtäimellä parempi tapa yksityisen pilven skaalaamiseen on horizontal scaling eli horisontaalinen skaalaus. Horisontaalisella skaalauksella tarkoitetaan uusien fyysisten palvelimien lisäämistä, jotta kuorma jakaantuu useiden laitteiden taakaksi. Horisontaalinen skaalaaminen on selvästi haastavampaa, kuin vertikaalinen skaalaus ja se luo ympäristöön lisää monimutkaisuutta. Vertikaalista skaalausta pidetään kuitenkin yleisesti lyhyemmän aikavälin hyötynä, koska kuorman jatkuvasti kasvaessa on horisontaalinen skaalaus joka tapauksessa tulossa eteen. (Phil. 2018)

Useimmissa julkisissa pilvissä puolestaan ei juurikaan tarvitse murehtia skaalauksen vuoksi, sillä suurin osa julkisista pilvistä on valmiiksi virtualisoitu, jolloin vertikaalista skaalausta voidaan toteuttaa melkein loputtomiin (Phil. 2018). Esimerkiksi Amazon Web Services tarjoaa palvelua Elastic Beanstalk, joka monitoroi sovelluksen käyttöastetta ja skaalaa resursseja ylös ja alas automaattisesti (Hansen. 2017.)

5.2 Infrastruktuurin perustamisen vaiva

OpenShiftissä koneoppimis- ja datatiederatkaisuiden asennus, konfigurointi ja käyttö on loppujen lopuksi todella kätevää ja helppoa, kunhan oikeat ratkaisut ja osat on aiemmin todettu yhteensopiviksi ja OpenShiftissä toimiviksi. Toki myös yhtä helppoa, ellei jopa helpompaa on käyttää jo valmista alustaa, kuten Google Cloud Platformia tai Amazon Web Serviceä. Näistä molemmissa on valmiiksi tarjolla olevia koneoppimisen ja datatieteen ratkaisuja, joita voi ottaa kuka tahansa käyttöön. Esimerkiksi Google Cloud Platformissa saa ilmaiseksi käyttöönsä 300 dollaria ensimmäiseksi 12 kuukaudeksi ja alustalta löytyy yli 25 koneoppimiseen ja datatieteeseen liittyvää työkalua (Products and services).

Oletetaan, että yksityistä pilveä ei ole vielä pystytetty ja ensimmäisenä vaihtoehtona on oman OpenShiftin pystyttäminen ja esimerkiksi TensorFlowin asentaminen ja käyttöönotto ja toisena vaihtoehtona kolmannen osapuolen alustan käyttäminen. Ensimmäinen vaihtoehto vaatii todella paljon vaivaa verrattuna toiseen vaihtoehtoon. Pelkästään jo OpenShiftin asentaminen yritystason tuotantokäyttöön on aikaa vievää ja paljon asiantuntemusta vaativa tehtävä. Käyttöönoton jälkeen ympäristö vaati myös jatkuvaa ylläpitoa ja esimerkiksi versiopäivityksiä. OpenShiftin ollessa asennettuna koneoppimis- ja datatiedeinfrastruktuurin perustaminen vaatii myös jonkin verran enemmän vaivaa, kun mitä se vaatisi kolmannen osapuolen alustaa käytettäessä.

Taloudelliset hyödyt oman yksityisen pilven käytöstä näkyvät vasta pitkällä tähtäimellä, mutta oman pilven käytöstä on myös muitakin hyötyjä, kuten ongelmatilanteissa tai palvelun käyttökatkoksissa valta ja ongelmien korjaus on kokonaan omissa käsissä. Kolmannen osapuolen palvelua käytettäessä käyttökatkoksille ei oikein voi tehdä muuta, kuin odottaa kärsivällisesti korjausta. Hyötynä on myös, että omassa pilvessä toteutettua ratkaisua on myös mahdollista tuotteistaa ja tarjota asiakkaille myytävänä pakettina. Toisaalta on myös mahdollista myydä kolmannen osapuolen alustan työkaluja palveluina, mutta niistä joudutaan aina maksamaan erilaisia käyttö- ja/tai lisenssimaksuja.

6 POHDINTA JA JOHTOPÄÄTÖKSET

Opinnäytetyön tulos ja hyöty toimeksiantajalle jäi hieman laihaksi, johtuen lähes kaikkien ratkaisuiden toimimattomuudesta toimeksiantajan OpenShift ympäristössä. Vielä tarkemmalla asiaan perehtymisellä ja ekstensiivisellä testaamisella, jolla ratkaisuiden dockerfile tiedostoa muokkaamalla yritetään saada ratkaisu olemaan käyttämättä root-oikeutta on kuitenkin lähes varmasti mahdollista saada ainakin osa ratkaisuista toimimaan toimeksiantajan OpenShiftissä. Dockerfilen muokkaaminen aiheuttaa aina sen, että docker image joudutaan rakentamaan uudestaan ja lataamaan Docker Hubin tai OpenShiftin rekisteriin, joka saattaa viedä jopa useita kymmeniä minuutteja verkkoyhteyksistä riippuen.

Kaikkien testattujen ratkaisuiden tulisi ajonaikaisesti toimia OpenShiftissä ilman root-oikeuksia, mutta docker imageiden koostajat ovat luoneet imaget niin, että asennuksen aikana sovellukset vaativat root-oikeudet.

Toimeksiantajan OpenShiftin käyttöoikeuksien tiukkuus oli jo ennen opinnäytetyön aloitusta tiedossa ja oikeuksien kiertämiseen ei saatu opinnäytetyössä kehitettyä ratkaisua. Käyttöoikeudet ovat tarkoituksenmukaisesti asetettu tiukoiksi, jottei mahdollisesti virheellinen tai vaarallinen sovellus pääse vakoilemaan tai tekemään tuhoa podissa, jossa se on ajossa. Asentaessa OpenShiftin lokaalisti ja vaihtamalla root-oikeuden käyttämisen sallituksi ratkaisuiden asentaminen ja käyttöönotto onnistui sujuvasti, joka takaa ratkaisuiden toimivuuden OpenShiftissä.

Opinnäytetyötä tehtäessä perehdyin OpenShiftiin, dockeriin, koneoppimiseen, datatieteeseen, pythoniin, TensorFlowiin ja Apache Sparkiin vähintään sillä tasolla, että kaikista jäi osaamiseksi hyvä perustaso.

Opinnäytetyön yhtenä tavoitteena oli, että olisi hyvä miettiä, kuinka koneoppimisen ja datatieteen ratkaisuita OpenShiftin päällä voitaisiin mahdollisesti tuotteistaa. Monien ratkaisuiden toimimattomuus toimeksiantajan OpenShiftissä aiheutti kuitenkin sen, ettei opinnäytetyössä jäänyt aikaa koostaa valmiita monien ratkaisuiden kokonaisuuksia, jotka olisi helposti tuotteistettavissa. Apache Sparkin toimiminen toimeksiantajan OpenShiftissä antoi kuitenkin toivoa ja sellaisen käsityksen, että monien ratkaisuiden kokonaisuuksia tai mahdollisesti yksittäisen ratkaisun tuotteistamista olisi mahdollista toteuttaa. Tuotteistamisen tueksi voisi luoda OpenShiftiin tietynlaisia valmiita templateja, joiden avulla

koneoppimisen ja/tai datatiede kokonaisuuden asentaminen onnistuisi aina mahdollisimman helposti.

Tuotteistettuja kokonaisuuksia olisi suhteellisen helppo tarjota valmiina paketteina asiakkaille. Asiakkaiden tarvitsisi vain toimittaa dataa, josta luotaisiin datatieteen ja koneoppimisen avulla esimerkiksi ennustettuja riskilukuja laitteen rikkoutumiselle ja jäljellä olevia käyttövuosia. Datatieteen ja koneoppimisen mahdolliset hyödyt ovat monilta yrityksiltä vieläkin tiedostamatta ja aiemmin toteutetuilla projekteilla ja valmiina olevilla tuotteistetuilla ratkaisuilla olisi hyvät mahdollisuudet vakuuttaa potentiaaliset asiakkaat.

LÄHTEET

Apache Spark Examples. Luettu 4.11.2018. https://spark.apache.org/examples.html

Get Started with the CLI. Luettu 22.9.2018. https://docs.openshift.com/enterprise/3.0/cli_reference/get_started_cli.html

Hansen, T. 2017. Unlimited Scaling on AWS. Luettu 15.11.2018. https://www.novataris.com/about-us/blogs/unlimited-scaling-on-aws/

Hope, T. & Resheff, Y, S. & Lieder, I. 2017. Learning TensorFlow. O'Reilly Media, Inc.

Lao, R. 2018. A Beginner's Guide to the Data Science Pipeline. Luettu 21.10.2018. https://towardsdatascience.com/a-beginners-guide-to-the-data-science-pipeline-a4904b2d8ad3

Liu, A. 2016. Apache Spark Machine Learning Blueprints. Packt Publishing.

Machine Learning Market. Luettu 24.9.2018. https://www.marketsandmarkets.com/Market-Reports/machine-learning-market-263397704.html#

Mohammed, M. & Khan, M, B. & Bashier, E, B. 2016. Machine Learning. CRC Press.

Nandi, A. 2016. Spark for Python Developers. Packt Publishing.

OpenShift Products. Luettu 23.9.2018. https://blog.openshift.com/category/products/

Ostrowski, R. Introduction to Apache Spark with Examples and Use Cases. Luettu 24.9.2018. https://www.toptal.com/spark/introduction-to-apache-spark

Phil. 2018. What is scalability in cloud computing? Luettu 14.11.2018. https://linuxacademy.com/blog/cloud/scalability-cloud-computing/

Pierson, L. & Porway, J. 2017. Data Science for Dummies. John Wiley & Sons, Incorporated.

Products and services. Luettu 15.11.2018. https://cloud.google.com/products/

Protasiewicz, J. 2018. Why Is Python So Good for AI, Machine Learning and Deep Learning? Luettu 4.11.2018. https://www.netguru.co/blog/why-is-python-so-good-for-ai-machine-learning-and-deep-learning

Provost, F. & Fawcett, T. 2013. Data Science for Business. O'Reilly.

Radanalytics. Luettu 22.9.2018. https://radanalytics.io/

Raj, P. & Singh, V. & Holla, S. & Khare, N. & Hane, O. & Konduri, S, K. & Kohli, V. & McKendrick, R. & Dua, R. & Chelladhurai, J, S. 2016. Docker: Creating Structured Containers. Packt Publishing.

SethBling. 2015. MarI/O – Machine Learning for Video Games. Katsottu 12.8.2018. https://www.youtube.com/watch?v=qv6UVOQ0F44

Shipley, G. & Dumpleton, G. 2016. OpenShift for Developers. O'Reilly Media, Inc.

Sinnett, W.M. 2010. Software As a Service : Experiences of SMBs. Morristown, N.J.

TensorFlowOnSpark. Luettu 18.11.2018. https://github.com/yahoo/TensorFlowOnSpark

Train your first neural network: basic classification. Luettu 29.10.2018. https://www.tensorflow.org/tutorials/keras/basic_classification

Watts, S. 2017. SaaS vs PaaS vs IaaS: What's The Difference and How To Choose. Luettu 12.8.2018. https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/