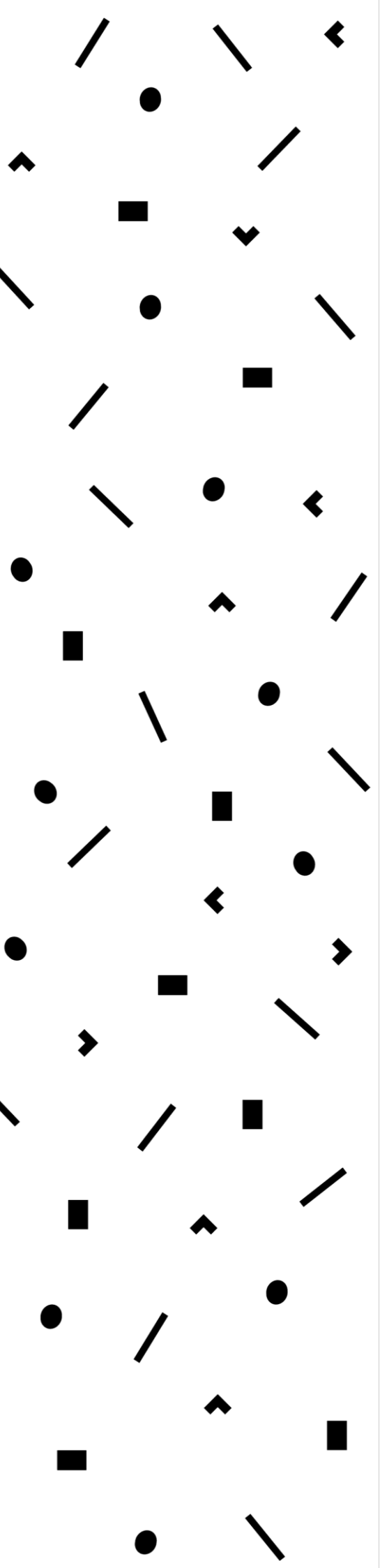


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

# HITSAUSTELEMETRIAN ANALYSOINTI JA TALLENNUS

Case: Kemppi Oy

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2018
Matias Pajuvesa

## Tiivistelmä


| Tekijä(t) | Julkaisun laji | Valmistumisaika |
|---|---|---|
| Pajuvesa, Matias | Opinnäytetyö, AMK | Kevät 2018 |
| | Sivumäärä | |
| | 43 | |


Työn nimi

**Hitsaustelemetrian analysointi ja tallennus**
Case: Kemppi Oy

Tutkinto

Tieto- ja viestintätekniikan koulutus


Tiivistelmä

Opinnäytetyön tarkoituksena oli kehittää toimiva prosessi WeldEye-pilvipalvelun hitsaustelemetrian vastaanottoon, analysointiin, tallennukseen ja analysoidun telemetriatiedon raportointiin. Smart Reader ja X8 MIG Welder IoT-laitteet lähettävät ja vastaanottavat tietoa WeldEye-pilvipalvelusta. Opinnäytetyön toimeksiantajana on Kemppi Oy.

Hitsaustelemetrian käsittelyn prosessi on toteutettu käyttäen Amazon Web Services -pilvilaskenta-alustaa. Amazon Web Services mahdollistaa skaalautuvien sovellusten kehityksen. Työssä on käytetty hyödyksi useita Amazon Web Servicesin tarjoamia pilvipalveluita. Näihin kuuluvat muun muassa EC2, Kinesis Data Stream, Lambda, S3 ja Elasticsearch. Palvelinpuolen logiikka on kirjoitettu Node.js-ajoympäristössä ja WeldEye-käyttöliittymä Angular-ohjelmistokehityksellä. Palvelinpuolen sovelluksen kehityksessä on käytetty hyödyksi Serverless- ja mikropalveluarkkitehtuureja. Prosessin oleellisia osia on sen luotettavuus ja skaalautuvuus. Mikropalveluarkkitehtuuri mahdollistaa uusien ominaisuuksien kehityksen WeldEye-pilvipalveluun nopeasti.

Työn tavoitteisiin päästiin ja lopputuloksena on toimiva ratkaisu hitsaustelemetrian vastaanottoon, analysointiin, tallennukseen ja analysoidun telemetriatiedon raportointiin. Työn toteutuksessa ei tullut esiin ylitsepääsemättömiä ongelmia. Suurimmat haasteet liittyivät sovelluksen skaalautuvuuteen, nopeuteen ja telemetriatiedon määrään.



Avainsanat

Amazon Web Services, Node.js, serverless, mikropalvelunarkkitehtuuri, skaalautuvuus

## Abstract


| Author(s) | Type of publication | Published |
|---|---|---|
| Pajuvesa, Matias | Bachelor's thesis | Spring 2018 |
| | Number of pages | |
| | 43 | |


Title of publication

**Analyzing and recording of welding telemetry**
Case: Kemppi Oy

Name of Degree

Bachelor's Degree Programme in Information and Communications Technology


Abstract

The objective of this thesis was to develop a viable process to receive, analyze, store and report welding telemetry data in the WeldEye cloud service. Smart Reader and X8 MIG Welder IoT devices send and receive information from WeldEye cloud service. The Bachelor's thesis was commissioned by Kemppi Oy.

The processing of welding telemetry was developed using Amazon Web Services platform. Amazon Web Services enables development of scalable software. This thesis uses many services provided by Amazon Web Services, including services such as EC2, Kinesis Data Stream, Lambda, S3 and Elasticsearch. Backend logic is written using Node.js runtime and the WeldEye user interface is written using the Angular framework. Backend software uses Serverless- and microservice architectures. The core aspects of the software are its reliability and scalability. Microservice architecture provides the possibility to develop new features quickly.

The goals of the work were achieved, and the result is a viable solution to receive, analyze, store and report welding telemetry data. There were no insurmountable problems. The biggest challenges were related to the application's scalability, speed and the amount of telemetry data.



Keywords

Amazon Web Services, Node.js, serverless, microservice architecture, scalability

SISÄLLYS

# 1 JOHDANTO

Opinnäytetyön toimeksiantaja on Kemppi Oy, joka on vuonna 1949 perustettu hitsausalan perheyritys. Kemppi kehittää älykkäitä laitteita, hitsaustuotannon hallintaohjelmistoja ja näitä tukevia asiantuntijapalveluita vaativiin teollisiin sovelluksiin kuin kuluttajan tarpeisiin. Kemppi työllistää yli 600 hitsauksen asiantuntijaa 13 maassa. Kempin liikevaihto oli yli 110 miljoonaa euroa vuonna 2017. (Kemppi Oy 2018a.) Kempin asiakkaisiin kuuluu telakka- ja offshore-teollisuus, rakennusteollisuus, putkien ja putkistojen valmistajat, auto- ja kuljetusteollisuus, konepajateollisuus ja useat muut teollisuusalat. (Kemppi Oy 2018b.)

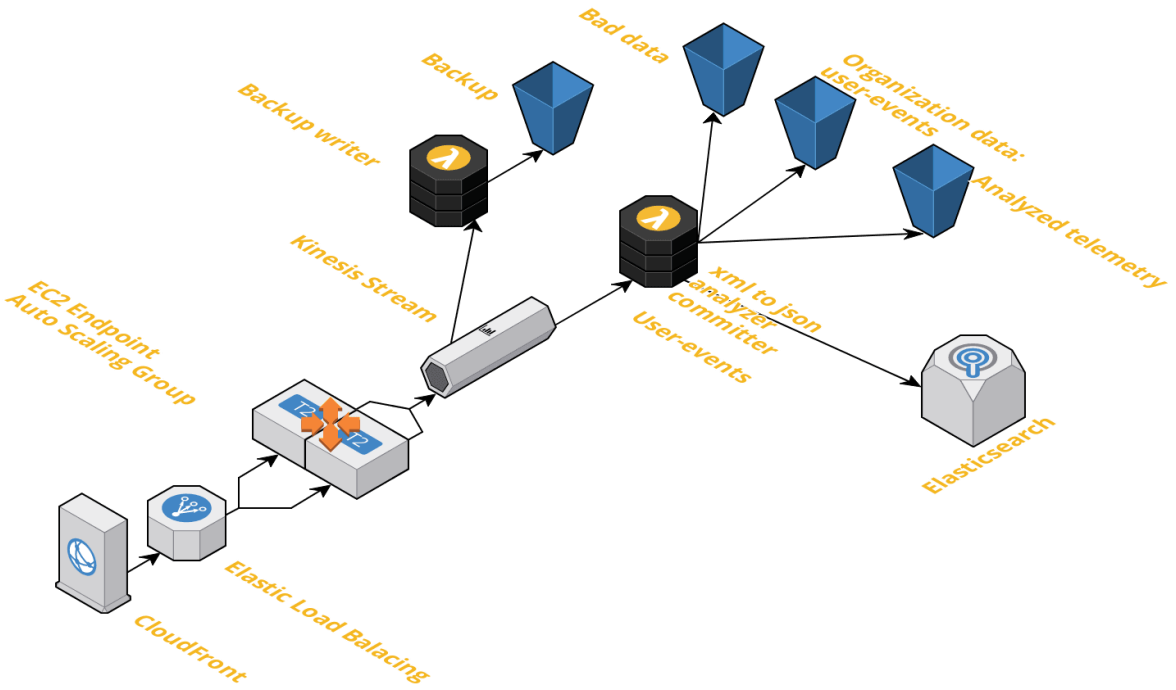
Opinnäytetyön tavoitteena on kehittää toimiva prosessi hitsaustelemetrian käsittelylle jatkuvasti kasvavassa pilvipalvelussa. Prosessiin kuuluu telemetriatiedon vastaanotto, analysointi, tallennus ja analysoidun telemetriatiedon raportointi. Prosessin pitää olla skaalautuva kasvavassa käyttäjäkunnassa.

Työ on osa WeldEye-pilvipalvelua. WeldEye on Kemppi Oy:n kehittämä pilvipalvelu hitsauksenhallintaan. WeldEye-pilvipalvelun tarkoitus on antaa universaali toteutus hitsauksenhallintaan. WeldEye sopii kaiken tyyppisille ja kokoisille hitsausta suorittaville yrityksille, jotka tarvitsevat kansainvälisiä ISO-, ASME-, AWS-standardeja. WeldEye tarjoaa ratkaisun koko prosessiin, hitsausohjeet, henkilöstön pätevyyksiin, dokumentaatioon, raportointiin ja hallintaan. Tärkeimmin WeldEye mahdollistaa 100% jäljitettävyyden jokaisesta hitsistä. (Weldeye 2018.)

Hitsauksen laadunvarmistuksessa on useita vaiheita. Ensinnäkin valmistajan tulee varmistua hitsaajiensa pätevyydestä – tarvittavat pätevöintimenetelmät on standardoitu. Hitsauksen laadunvarmistamisen apuna toimivat myös yhtenäiset kirjalliset hitsausohjeet (Welding Procedure Specification/WPS), joiden tulee myös olla hyväksytyt. Tähän on käytössä useita standardoituja menettelyjä. (Inspecta 2018.)

# 2 TOIMINTAKUVAUS

WeldEye on toteutettu Amazon Web Services (AWS)-pilvilaskenta-alustan palveluilla. Amazon Web Services tarjoaa laajan kokoelman pilvipohjaisia palveluita. Tässä työssä on käytetty monia Amazon Web Servicesin tarjoamia pilvipalveluita. Näihin kuuluvat muuan muassa Amazon EC2, Amazon Kinesis Data Stream, AWS Lambda, Amazon S3, Amazon Elasticsearch Service. Kuviossa 1 on esitetty suunnitelma hitsaustelemetrian prosessoinnin arkkitehtuurista.




KUVIO 1. Suunnitelma käsittelyprosessin arkkitehtuurista

Hitsaustelemetrian käsittelyprosessin arkkitehtuuri pitää olla toteutettu siten, että se on hyvin skaalautuva. Sovelluksen skaalautumisessa pitää ottaa huomioon samanaikaisten kyselyiden määrä sekunnissa, keskimääräinen vasteaika pyynnössä ja käsiteltyjen tuetuiden määrä sekunnissa/minuutissa (StackExchange 2018). Käsittelyprosessin pitää olla skaalautuva käsittelemään satoja pyyntöjä minuutissa. Käsittelyprosessin arkkitehtuuri on toteutettu siten, että osia prosessista voidaan muokata tai vaihtaa ilman, että koko prosessia pitää suunnitella tai toteuttaa uudelleen. Amazon Web Servicesin pilvipalveluita käyttämällä prosessin toteutus on mahdollista kasvavassa käyttäjäkunnassa.

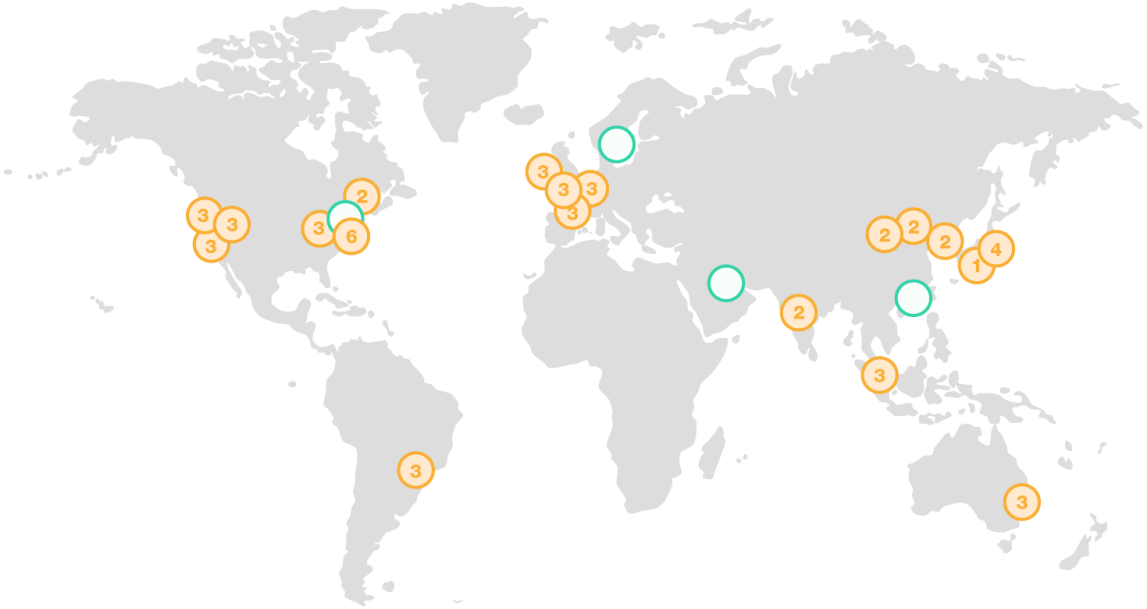
WeldEye-pilvipalvelun käyttöliittymä on toteutettu Angular-kirjastolla. Angular on Googlen kehittämä avoimen lähdekoodin JavaScript-ohjelmistokehys. Angular-ohjelmistokehystä ei tule sekoittaa AngularJS-ohjelmistokehykseen. Angular on kirjoitettu kokonaan uudelleen AngularJS-ohjelmistokehyksestä.

Sovelluksen käyttöliittymä pyritään toteuttaa kehittämällä mahdollisimman pieniä komponentteja ja moduuleja, joka mahdollistaisi sovelluksen jatkokehityksen ja ylläpidon mahdollisimman sulavasti.

# 3 KÄYTETYT TEKNOLOGIAT

WeldEye-pilvipalvelu ja siihen liittyvät sovellukset on toteutettu Amazon Web Services palveluilla. Amazon Web Services tarjoaa laajan kokoelman globaaleja pilvipohjaisia tuotteita. Näihin tuotteisiin sisältyy muun muassa laskenta-, tietokanta-, analytiikka-, verkko-, mobiili-, kehittäjä-, hallinta-, IoT-, turvallisuus- ja yrityssovelluksia. (AWS 2018f.) Amazon Web Services tarjoaa sovelluskehittäjille laajan valikoiman valita sopivan palvelun sovelluskehityksessä.

Amazon Web Services tarjoaa saatavuuden 54 alueella, 18 maantieteellisellä alueella ja 1 paikallisella alueella. Laaja saatavuusalueiden määrä tarjoaa AWS-palveluiden asiakkaille helpomman ja tehokkaamman tavan suunnitella ja operoida sovelluksia ja tietokantoja, tekemällä ne enemmän käytettävimmiksi, vikasietoisemmaksi, ja skaalautuvammaksi kuin yksittäinen palvelinkeskus tai monen palvelinkeskuksen infrastruktuuri. (AWS 2018d.)




KUVIO 2. Amazon Web Services infrastruktuuri (AWS 2018d)

Kuviossa 2 ympyrät merkitsevät maantieteellisiä alueita ja numerot saatavuusalueita. Tulevat alueet on merkitty vihreällä. Kuviosta nähdään, että Amazon Web Services tarjoaa laajan kattavuuden ympäri maailmaa.

Amazon Web Services tarjoaa sovellusten ja tiedon replikointia monelle palvelinkeskukselle. Kehittäjä voi myös lisätä redundanssia ja vikasietoisuutta replikoimalla tiedon maantieteellisten alueiden välillä (AWS 2018d). Palvelinkeskusten välinen replikointi mahdollistaa sovelluksen toimimisen esimerkiksi palvelinkeskukseen kohdistuvan vian takia.

Mannertenvälinen replikointi mahdollistaa taas sovelluksen toimisen esimerkiksi merenpohjaan laskettujen tietoliikennekaapelien katkeamisen yhteydessä.

Amazon Web Services palveluita käyttää moni tunnettu yritys. Näihin yrityksiin kuuluu muun muassa: Supercell, Spotify, Siemens, Adobe, Philips ja Rovio (AWS 2018g).

Amazon Web Services tarjoaa ilmaisen käyttäjätason laajalla sovellusmäärällä ja käyttötasolla. Palvelut mahdollistavat sovelluksen skaalautumisen helposti käyttäjämäärien noustessa. Palveluiden maksuperiaate perustuu myös sovelluksien käyttöaikaan. Palveluiden käyttö poistaa sovellusten kehittäjiltä vaatimuksen ostaa ja hallita fyysisiä palvelimia. Fyysisien palveluiden osto vaatimuksen poisto edesauttaa nopeampaa ja halvempaa sovelluskehitystä. Ilmainen käyttäjätaso mahdollistaa palveluiden testaamisen ja käyttämisen yksittäiselle tai pienelle kehittäjätiimille pienellä laskulla.

**Compute**
Amazon EC2
Amazon EC2 Auto Scaling
Amazon Elastic Container Service
Amazon Elastic Container Service for Kubernetes
Amazon Elastic Container Registry
Amazon Lightsail
AWS Batch
AWS Elastic Beanstalk
AWS Fargate
AWS Lambda
AWS Serverless Application Repository
Elastic Load Balancing
VMware Cloud on AWS

**Storage**
Amazon Simple Storage Service (S3)
Amazon Elastic Block Storage (EBS)
Amazon Elastic File System (EFS)
Amazon Glacier
AWS Storage Gateway
AWS Snowball
AWS Snowball Edge
AWS Snowmobile

**Database**
Amazon Aurora
Amazon RDS
Amazon DynamoDB
Amazon ElastiCache
Amazon Redshift
Amazon Neptune
AWS Database Migration Service

**Migration**
AWS Migration Hub
AWS Application Discovery Service
AWS Database Migration Service
AWS Server Migration Service
AWS Snowball
AWS Snowball Edge
AWS Snowmobile

**Networking & Content Delivery**
Amazon VPC
Amazon CloudFront
Amazon Route 53
Amazon API Gateway
AWS Direct Connect
Elastic Load Balancing

**Developer Tools**
AWS CodeStar
AWS CodeCommit
AWS CodeBuild
AWS CodeDeploy
AWS CodePipeline
AWS Cloud9
AWS X-Ray
AWS Tools & SDKs

**Management Tools**
Amazon CloudWatch
AWS Auto Scaling
AWS CloudFormation
AWS CloudTrail
AWS Config
AWS OpsWorks
AWS Service Catalog
AWS Systems Manager
AWS Trusted Advisor
AWS Personal Health Dashboard
AWS Command Line Interface
AWS Management Console
AWS Managed Services

**Media Services**
Amazon Elastic Transcoder
Amazon Kinesis Video Streams
AWS Elemental MediaConvert
AWS Elemental MediaLive
AWS Elemental MediaPackage
AWS Elemental MediaStore
AWS Elemental MediaTailor

**Machine Learning**
Amazon SageMaker
Amazon Comprehend
Amazon Lex
Amazon Polly
Amazon Rekognition
Amazon Machine Learning
Amazon Translate
Amazon Transcribe
AWS DeepLens
AWS Deep Learning AMIs
Apache MXNet on AWS
TensorFlow on AWS

**Analytics**
Amazon Athena
Amazon EMR
Amazon CloudSearch
Amazon Elasticsearch Service
Amazon Kinesis
Amazon Redshift
Amazon QuickSight
AWS Data Pipeline
AWS Glue

**Security, Identity & Compliance**
AWS Identity and Access Management (IAM)
Amazon Cloud Directory
Amazon Cognito
Amazon GuardDuty
Amazon Inspector
Amazon Macie
AWS Certificate Manager
AWS CloudHSM
AWS Directory Service
AWS Key Management Service
AWS Organizations
AWS Single Sign-On
AWS Shield
AWS WAF
AWS Artifact

**Mobile Services**
AWS Mobile Hub
Amazon API Gateway
Amazon Pinpoint
AWS AppSync
AWS Device Farm
AWS Mobile SDK

**AR & VR**
Amazon Sumerian

**Application Integration**
Amazon MQ
Amazon Simple Queue Service (SQS)
Amazon Simple Notification Service (SNS)
AWS AppSync
AWS Step Functions

**Customer Engagement**
Amazon Connect
Amazon Pinpoint
Amazon Simple Email Service (SES)

**Business Productivity**
Alexa for Business
Amazon Chime
Amazon WorkDocs
Amazon WorkMail

**Desktop & App Streaming**
Amazon WorkSpaces
Amazon AppStream 2.0

**Internet of Things**
AWS IoT Core
Amazon FreeRTOS
AWS Greengrass
AWS IoT 1-Click
AWS IoT Analytics
AWS IoT Button
AWS IoT Device Defender
AWS IoT Device Management

**Game Development**
Amazon GameLift
Amazon Lumberyard

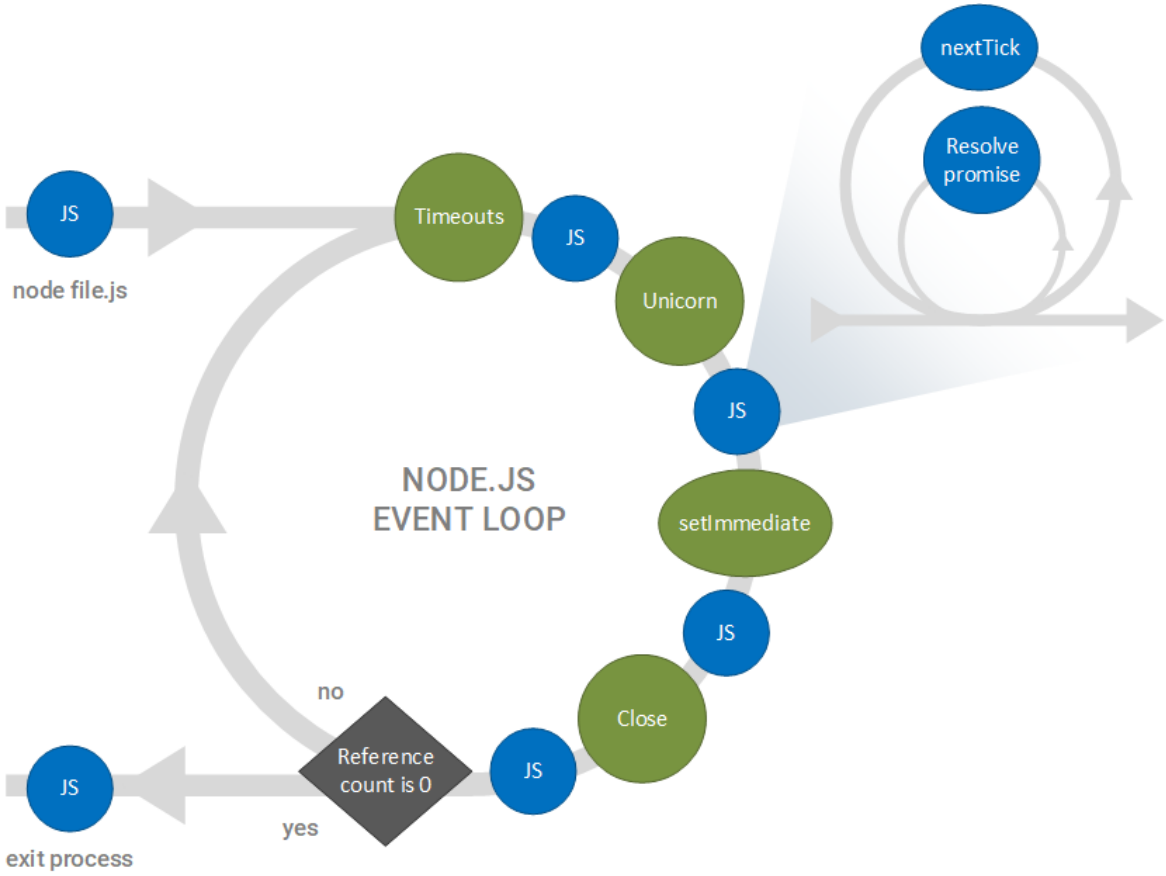
**Software**
AWS Marketplace

**AWS Cost Management**
AWS Cost Explorer
AWS Budgets
Reserved Instance Reporting
AWS Cost and Usage Report

KUVIO 3. Lista Amazon Web Services -palveluita (AWS 2018f)

## 3.1 Node.js

Node.js on JavaScript ajoympäristö Chromen V8 JavaScript-moottorilla. Node.js käyttää tapahtumapohjaista ei-blokkaavaa I/O-mallia. (Node.js 2018b.)




KUVIO 4. Node.js tapahtumapohjainen malli (Dzone / Web Dev Zone 2018)

Asynkronisen tapahtumapohjaisen I/O-mallin ansiosta Node.js sopii hyvin skaalautuviin webpohjaisiin toteutuksiin. Monta samanaikaista kyselyä voidaan suorittaa ja jokaisen kyselyn kohdalla, suoritetaan takaisinkutsu. (Node.js 2018a.) Kuviossa 4 esitetään Node.js tapahtumapohjainen malli.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

KUVIO 5. Node.js ”Hello World”-websovellus (Node.js 2018a)

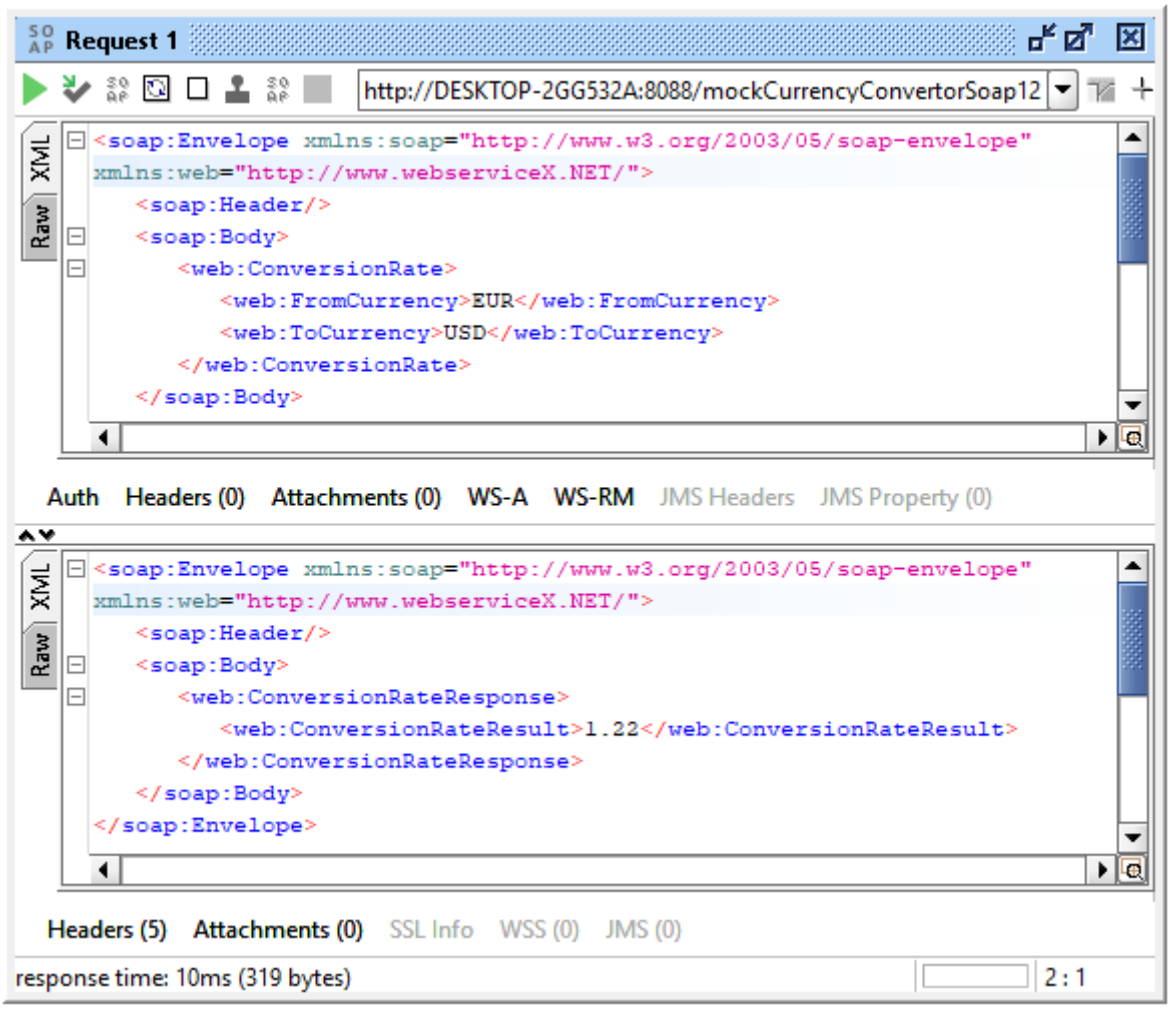
Node.js tapahtumapohjainen malli on vastoin nykyajan yleisempää samanaikaisuuden mallia, jossa käytetään säikeitä. Säie-pohjainen verkkotyöskentely on suhteellisen tehotonta ja vaikeaa käyttää. Lähes mikään funktio Nodessa ei suoraan suorita I/O-operaatioita, joten prosessi ei ikinä odota. Koska mikään ei ikinä odota, skaalautuvat järjestelmät ovat järkeviä toteuttaa Nodella. (Node.js 2018a.) Kuviossa 5 esitetään websovellus. Kaikki palvelinpuolen sovelluskoodi on kirjoitettu Node.js-ajoympäristössä.

## 3.2 Rajapinnat

Telemetrian vastaanotto vaatii kahta rajapintaa. Ensimmäinen näistä on SOAP. Aikaisemmin kehitetty Smart Reader -laite käyttää SOAP-rajapintaa kommunikointiin. Toinen rajapinta on REST. REST-rajapintaa käyttää X8 MIG Welder ja mahdolliset tulevat laitteet.

### 3.2.1 SOAP

SOAP on protokolla, joka on tarkoitettu jäsenneltyjen tietojen vaihtamiseen hajautetussa ympäristössä. SOAP käyttää XML-dokumenttia määrittämään laajennettavissa olevan viestikehyksen, joka voidaan lähettää erilaisten taustaprotokollien ylitse. (W3C 2018.)




KUVIO 6. Esimerkki SOAP pyyntö ja vastaus

Kuviossa 6 esitetään esimerkkipyyntö ja vastaus SOAP-tekniikalla. Standardin mukaan SOAP-viesti koostuu kolmesta osasta: pakollisesta SOAP-envelope, otsikkotietoja sisältävästä valinnaisesta SOAP-header:stä ja itse datan sisältävästä, pakollisesta SOAP-body:stä (VirtuaaliAMK 2018).

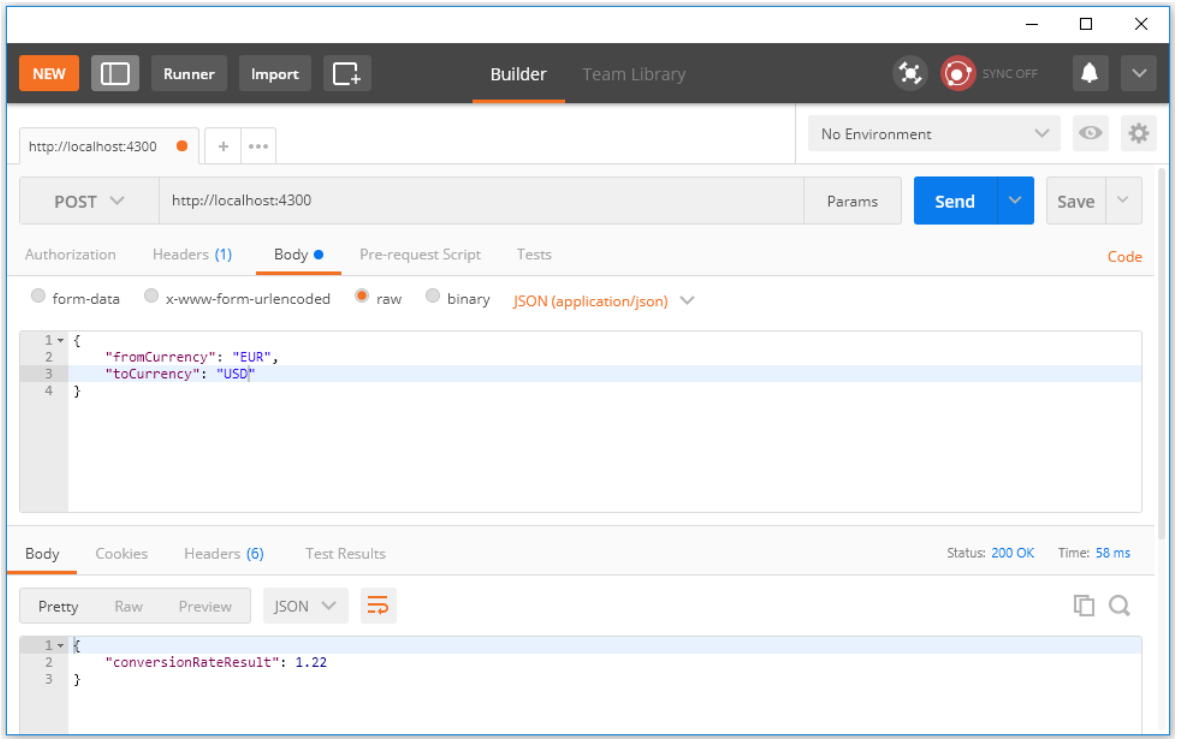
Envelope määrittelee välitetyn viestin rakenteen, eli mitä dataa viesti sisältää ja kuinka tuota dataa voidaan käsitellä (VirtuaaliAMK 2018). SOAP-envelope:n ansiosta SOAP-viestin rakenne on tarkasti määritelty. Tarkka määrittely avustaa rajapinnan oikeassa käytössä, koska käyttäjä voi lähettää ja vastaanottaa tietoa vain tietyssä muodossa. Tarkka rajapinnan määrittely vaikeuttaa myös rajapinnan päivitystä. Rajapintaa jatkokehitettäessä esimerkiksi uuden kentän lisäyksen kohdalla, pitää tehdä uusi versio rajapinnasta. Tällöin rajapinnan päivitys ei riko muita palveluita, jotka käyttävät vanhempaa versiota.

Encoding rules määrittää eri datatyyppien esitysmuodot. Header on yleinen mekanismi lisätä hajautetusti ominaisuuksia itse SOAP-viestiin, joka toimii riippumatta kommunikoivista laitteista. (VirtuaaliAMK 2018.)

RPC Represention, Remote Procedure Call, eli etäproseduurin kutsu. Etäolioiden metodeihin kohdistuvien kutsujen viestimuotoinen esitys ja vastaanotto. Body eli SOAP-viestin sisältö, jossa on hajautetusti säilöttynä viestin informaatio. (VirtuaaliAMK 2018.)

### 3.2.2 REST

REST (REpresentational State Transfer) on arkkitehtuurinen tyyli kehittää webpalveluita. REST on suosittu yksinkertaisuutensa ja sen takia, että se on kehitetty nykyisten järjestelmien ja ominaisuuksien kuin, että olisi kehitetty uusia standardeja, kirjastoja ja teknologioita. (TechTarget 2018.)




KUVIO 7. Esimerkki REST pyyntö ja vastaus

REST-arkkitehtuurissa käyttäjäpuolen ja palvelimen toteutus voidaan tehdä itsenäisesti ilman, että kumpikaan ei tiedä toisistaan. Tämä tarkoittaa sitä, että käyttäjäpuolen toteutus voidaan muuttaa milloin tahansa ilman vaikutusta palvelimeen, ja palvelimen toteutus voidaan vaihtaa ilman vaikutusta käyttäjäpuoleen. (codeacademy 2018.)

Tämä mahdollistaa sen, että käyttäjäpuolen ja palvelimen kehitykset voidaan toteuttaa samaan aikaan, kehittäjien pitää vain sopia, missä formaatissa tietoa lähetään ja vastaanotetaan.

Käyttäjäpuolen ja palvelimien eri toteutukset mahdollistavat myös toisen toteutuksen täyden uudelleenkirjoituksen ilman vaikutusta toiseen. Esimerkiksi sovelluksen käyttöliittymä voidaan kirjoittaa täysin uusiksi uudella tekniikalla, käyttäen vanhaa palvelinpuolta.

## 3.3 EC2

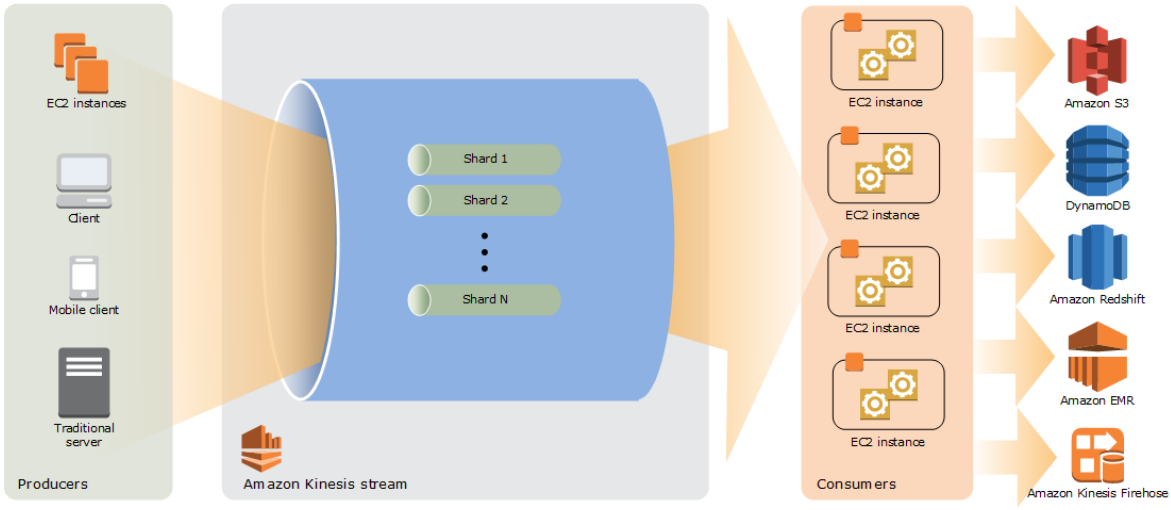
Amazon Elastic Compute Cloud (Amazon EC2) tarjoaa skaalautuvaa laskentatehoa Amazon Web Services (AWS) -pilvessä. EC2-virtuaalikoneet eliminoivat tarpeen sijoittaa palvelinlaitteistoon. Kehittäjä voi käynnistää niin monta virtuaalista palvelinta kuin tarvitsee. Amazon EC2 skaalautuu vaatimusten tai suosion mukaan, ilman verkkoliikenteen ennustamista. (AWS 2018i.)

EC2-virtuaalipalvelimet mahdollistavat palvelimien käytön pohjimmiltaan samalla tavalla, kuin fyysiset palvelimet toimistossa. Kehittäjä voi valita vaadittavan palvelinlaitteiston, käyttöjärjestelmän ja asentaa halutut sovellukset virtuaalipalvelimelle.

Amazon Web Services tarjoaa neljä eri tapaa maksaa EC2-virtuaalipalvelimista: Tarpeen vaatiessa (On-Demand), varatut instanssit (Reserved Instances), spot-instanssit (Spot Instances) ja Dedikoitu palvelin (Dedicated host). EC2-virtuuaalipalvelimien laskutus hoidetaan eri tavalla riippuen valitusta instanssin tyypistä. (AWS 2018a.)

## 3.4 Kinesis Data Stream

Kinesis Data Stream sisältää yhden tai useamman sirpaleen (Shard). Jokaisella sirpaleella on tietueiden sarja. Jokaisella tietuella on sekvenssinumero, jonka Kinesis-virta on antanut. (AWS 2018j.)




KUVIO 8. Esimerkki korkean tason arkkitehtuurikuvaus kinesis virrasta (AWS 2018j)

Sirpale on yksilöllisesti tunnistettu tietueiden sarja virrassa. Virta voi sisältää yhden tai useamman sirpaleen, ja jokainen sirpale tarjoaa tietyn määrän kapasiteettia. Yksittäinen sirpale tukee 5 lukutransaktiota sekunnissa ja maksimissaan 2 MB sekunnissa. Sirpale mahdollistaa maksimissaan 1000 tietueen ja 1 MB kirjoitusnopeuden. Virran kokonaiskapasiteetti on sirpaleitten yhteenlaskettu kapasiteetti. (AWS 2018j.) Sirpaleitten lisääminen tai vähentäminen näin mahdollistaa Kinesis-palvelun horisontaalisen skaalautumisen.

Osioavain (partition key) on avain, jolla ryhmitellään tietoa sirpaleisiin Kinesis-virrassa. Osioavain määritetään sovelluksessa, joka lähettää tietoa virtaan. Tuottaja (Producer) laittaa tietoa Kinesis-virtaan. Tuottajan pitää antaa Kinesis-virran nimi, osioavain ja tietue mikä lähetään. Kuluttaja (Consumer) lukee tietueita sirpaleesta käyttäen sirpale iteraattoria (shard iterator). Iteraattori määrittää sijainnin virrassa, mistä kuluttaja lukee tietueita. (AWS 2018j.)

Säilytysjakso (retention period) mahdollistaa tietueiden säilyttämisen virrassa 24:stä tunnista 168:aan tuntiin (AWS 2018j). Tämä mahdollistaa tiedon säilymisen mahdollisissa ongelmatilanteissa kuluttajasovelluksissa.

## 3.5 Lambda

AWS Lambda on palvelu, jossa voi ajaa ohjelmakoodia ilman palvelimia. Lambda ajaa ohjelmakoodin vain silloin, kun tarvitsee. Lambda skaalautuu automaattisesti muutamasta kyselystä päivässä tuhansiin sekunnissa. Lambda tukee tällä hetkellä Node.js, Java. C#, Go ja Python -ohjelmointikieliä. (AWS 2018k.)

AWS Lambda mahdollistaa serverless-arkkitehtuurin käytön palvelinpuolen ohjelmakoodissa. Serverless-arkkitehtuuri tarkoittaa sitä, että sovelluksen omistajan ei tarvitse ostaa, lainata tai vuokrata palvelimia tai virtuaalikoneita palvelinpuolen sovellukselle (Medium 2018). Serverless-arkkitehtuuri täten mahdollistaa palvelinpuolen sovellusten toteutuksen ilman fyysisen infrastruktuurin hankintaa ja hallinnointia.

AWS Lambda mahdollistaa myös mikropalveluarkkitehtuurin käytön palvelinpuolen logiikassa. Lambda sopii hyvin mikropalveluarkkitehtuuriin toimintaperiaatteensa takia, koska ohjelmakoodi ajetaan vain silloin, kun tarvitsee.

Lambdan laskutus perustuu lambdafunktioiden kutsumäärään ja suoritusaikaan. Amazon Web Services tarjoaa joka kuukausi miljoona lambdafunktiokutsua ja 400 000 gigatavusekuntia ilmaiseksi. Määrittämällä lambdafunktio käyttämään 512 Mt, käytössä on 800 000 gigatavusekuntia. (AWS 2018e.)

Mikropalvelut ovat arkkitehtuurisia ja organisatorisia lähestymistapoja ohjelmistokehitykseen, jossa ohjelmisto koostuu pienistä itsenäisistä palveluista, jotka kommunikoivat hyvin määriteltyjen ohjelmistorajapintojen ylitse. (AWS 2018h.) Mikropalvelujen pienen koon ja lyhyen suoritusajan ansioista mikropalveluarkkitehtuuri sopii hyvin lambdaan.

Mikropalvelujen arkkitehtuurimallin ansioista ohjelmiston skaalautuminen on helpompaan ja halvempaa kuin suuremmissa, ”monoliittisissa”-sovelluksissa. Yksittäisen mikropalvelun kapasiteettia voidaan nostaa, jos kapasiteetti loppuu kesken. AWS Lambda tukee tätä automaattisesti.

## 3.6 S3

Amazon S3 on tallennuspaikka objektipohjaisille tiedostoille. S3 mahdollistaa tiedon hakemisen ja tallentamisen mistä tahansa riippumatta tiedon määrästä. Amazon S3 toimii maailman suurimman globaalin pilvi-infrastruktuurilla. Tiedostot jaetaan automaattisesti minimissään vähintään kolmelle eri paikalle, jotka ovat maantieteellisesti samassa paikassa. Amazon S3 voi myös automaattisesti replikoida tiedon myös mille tahansa muulle AWS-alueelle. (AWS 2018c.) Amazon S3 on tästä johtuen erittäin vikasietoinen ei haluttua tiedonmenetystä vastaan.

Amazon S3 -palvelun laskutus perustuu käyttöön. Ensimmäiset 50 teragigaa kuukaudessa maksaa $0.023 dollaria gigaa kohden. Seuraavat 450 teragigaa kuukaudessa maksaa $0.022 per giga. Yli 500 teragigaa kuukaudessa maksaa $0.021 dollaria per giga. (AWS 2018c.)

PUT-, COPY-, POST- tai LIST-pyynnöt maksavat $0.005 dollaria per 1000 pyyntöä. GET ja muut pyynnöt maksavat $0.0004 dollaria per 1000 pyyntöä. Amazon tarjoaa uusille käyttäjille ilmaiseksi jokainen kuukausi 20 000 GET pyyntöä, 2000 PUT pyyntöä ja 15 gigaa tiedonsiirtoa ulospäin per kuukausi yhdeksi vuodeksi. (AWS 2018c.) S3-palvelun käyttö vuoden ajan pienessä käytössä on halpa tapa esimerkiksi varmuuskopioida tietoa.

## 3.7 Elasticsearch

Elasticsearch avoimeen lähdekoodiin perustuva hajautettu RESTful-pohjainen etsintä- ja analytiikkamoottori. Elasticsearch skaalautuu erinomaisesti mahdollistaen erittäin suuren määrän kyselyitä per sekunti. (elastic 2018b.)

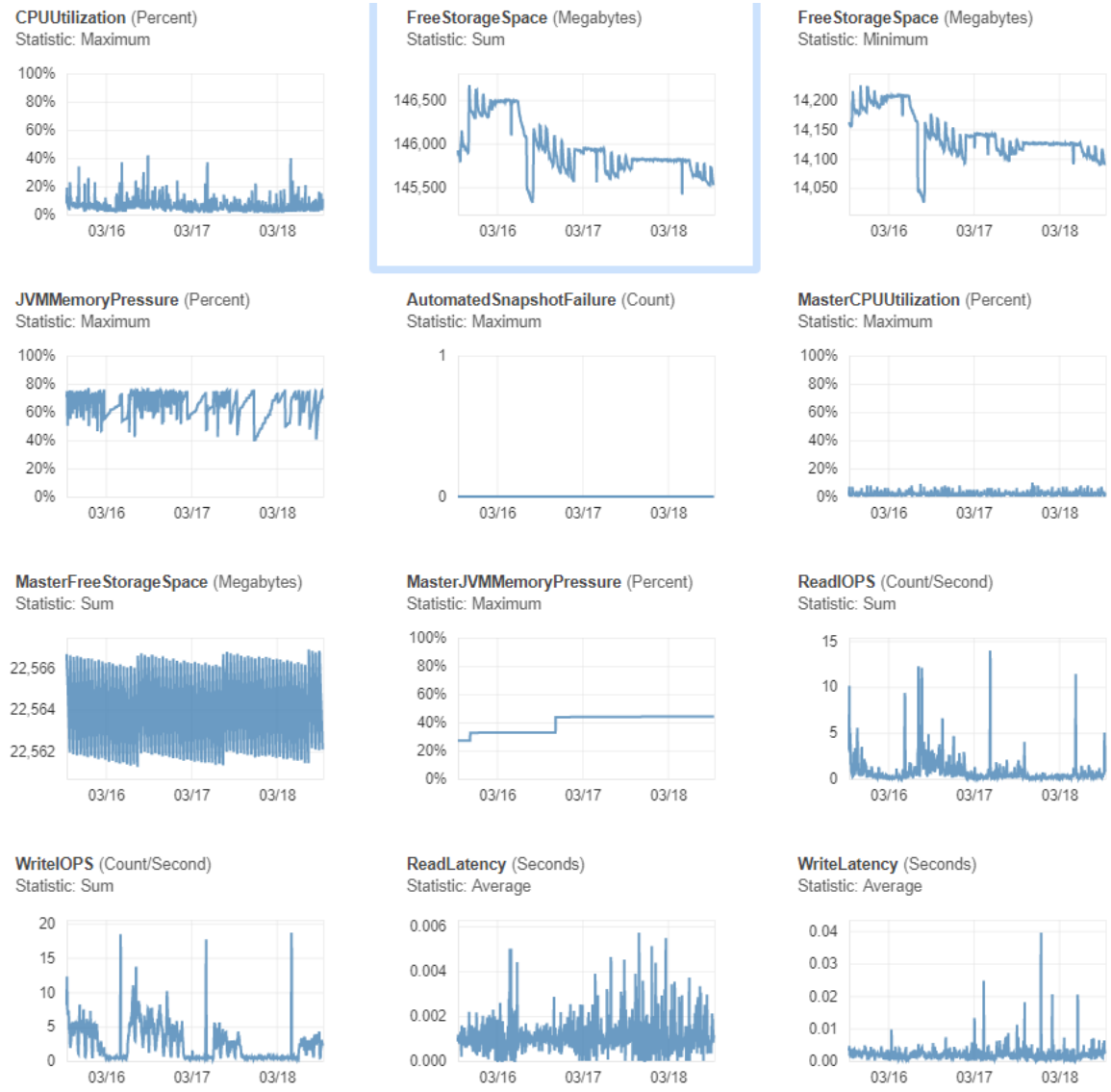
Klusteri (Cluster) on kokoelma yhdestä tai useammasta noodista (palvelimista), jotka hallitsevat tallennettua tietoa. Klusterit mahdollistavat indeksointi ja etsintämahdollisuudet kaikista noodeista. (elastic 2018a.)

Noodi (Node) on yksittäinen palvelin, joka on osa klusteria. Noodi tallentaa käyttäjän tiedon ja osallistuu klusterin indeksointi- ja etsintäkykyihin. (elastic 2018a.)

Indeksi (Index) on kokoelma dokumenteista, jotka ovat jokseenkin samanlaisia. Esimerkiksi kehittäjällä voi olla indeksi käyttäjätiedoista, tuotekatalogista ja tilaustiedoista. Indeksi identifioidaan nimen perusteella ja tätä nimeä käytetään dokumenttien indeksointiin, etsintään, päivityksiin ja poisto-operaatioihin. Dokumentti on yksittäinen elementti, joka voidaan indeksoida. Käyttäjä voi tallentaa halutun määrän dokumentteja indeksiin. (elastic 2018a.)

Sirpaleet (Shard) mahdollistavat indeksin jakamisen useampaan osioon. Yksittäinen indeksi voi esimerkiksi olla yli yhden teratavun kokoinen, ja ne eivät täten mahdu palvelimen levylle, tai etsintä olisi liian hidasta yhdeltä noodilta. Sirpaleet mahdollistavat operaatioiden horisontaalisen ja rinnakkaisen skaalautumisen. (elastic 2018a.)

Amazon Web Services tarjoaa Elasticsearch-palvelua. Elasticsearch toimii käytännössä identtisesti AWS-palvelussa. Amazon tarjoaa käyttöliittymän Elasticsearch indeksien luonnille ja muokkaamiselle ja monitoroinnille.




KUVIO 9. AWS Elasticsearch monitorointityökaluja

# 4 TELEMETRIATIEDON TALLENNUS JA ANALYSOINTI

## 4.1 Tavoite

Tämän toteutuksen tavoitteena on kehittää käsittelyprosessi hitsaustelemetrian vastaanottoon, käsittelyyn, tallennukseen ja raportointiin. Käsittelyprosessin pitää olla hyvin skaalautuva ja luotettava. Käsittelyprosessin pitää olla helposti jatkokehitettävissä uusilla ominaisuuksilla.

## 4.2 IoT-laitteet

IoT-laitteisiin kuuluu opinnäytetyön toteutuksen aikana Smart Reader- ja X8 MIG Welder -laitteet. Smart Reader sopii käytettäväksi WeldEye-ohjelmiston kanssa. Smart Reader toimii hitsaajan viivakoodinlukijana ja viestintävälineenä. Kuvassa 10 näytetään Smart Reader-laite. Se kerää digitaalista hitsausdataa työasemista ja siirtää sen WeldEye-ohjelmistoon. (Kemppi Oy 2018c.)


KUVA 10. Smart Reader

X8 MIG Welder tarjoaa kaiken, mitä tarvitaan synergisessä ja pulssi-MIG/MAG-hitsauksesta puikkohitsaukseen, MIG-kaarijuottoon, hiilikaaritalttaukseen ja pinnoitushitsaukseen.

Virtalähde takaa erittäin tarkan valokaaren hallinnan ja suorituskyvyn aina 600 ampeeriin asti. Järjestelmä on myös mahdollista yhdistää pilvipohjaiseen WeldEye-hitsaushallintaohjelmistoon. (Kemppi Oy 2018d.)


KUVIO 11. X8 MIG Welder (Kemppi Oy 2018d)

## 4.3 Vastaanottopalvelin

Vastaanottopalvelin on toteutettu käyttäen EC2-virtuaalipalvelinta. Palvelimen käyttöjärjestelmä on Linux-pohjainen. Palvelimella ajettava sovellus on toteutettu käyttäen Node.js-ajoympäristöä.

Vastaanottopalvelimen päivitys on toteutettu automaattisesti jatkuvan toimituksen -työkalujen avulla. Elastic Load Balancing- ja Auto Scaling Group -palvelut mahdollistavat vastaanottopalvelimen vikasietoisuuden ja skaalautumisen käyttäjämäärien kasvaessa.

Vastaanottopalvelimen vastuulla on toimia välikätenä IoT-laitteiden ja taustajärjestelmien välillä. Vastaanottopalvelimen toimintoihin kuuluu lähettää ja vastaanottaa IoT-laitteilta tietoa.

## 4.4 Kommunikointi IoT-laitteiden kanssa

Palvelimeen vaadittiin kahta eri rajapintaa tiedon vastaanottamiseen ja lähettämiseen. Aikaisemmin kehitetty Smart Reader -laite käyttää SOAP-rajapintaa kommunikointiin palvelimen kanssa. X8 MIG Welder ja muut tulevat IoT-laitteet käyttävät taas kommunikointiin nykyisemmin yleisempää REST-rajapintaa.

IoT-laitteet ottavat yhteyttä palvelimeen ajastetusti vastaanottaakseen organisaatiokohtaisia tietoja ja asetuksia. Jokainen pyyntö palvelimen REST- tai SOAP-rajapintaan autentikoidaan. Autentikoinnilla varmistetaan, että jokainen rajapintaan saapunut pyyntö saapuu oikealta laitteelta. Autentikoinnilla saadaan selville, mihin organisaatioon IoT-laite on liitetty, tällöin laitteella voidaan lähettää takaisin oikean organisaation tiedot. Organisaatiokohtaisiin tietoihin kuuluu esimerkiksi: hitsausohjeita, käyttäjiä, pätevyyksiä, lisäainelankoja, suojakaasuja ja hitsauslistoja.

Organisaatiotietojen vastaanoton jälkeen IoT-laitteella voidaan joko lukea viiva- tai QR-koodi tai syöttää esimerkiksi tunniste IoT-laitteelta. Hitsaajan lukiessa esimerkiksi hitsausohjeen viivakoodin laite tunnistaa hitsausohjeen ja näyttää sen laitteen näytöllä. Hitsaaja voi näin lukea myös muita tarvittavia viivakoodeja.


KUVA 12. Hitsaajan valitsemia viivakoodeja X8 Control Padissä

Kun hitsaajan aloittaa hitsaamisen, laite aloittaa keräämään telemetriatietoa muun muassa hitsausjännitteestä, virrasta ja lisäainelangansyötöstä. Telemetriatietoon lisätään hitsaajan lukemat viiva- tai QR-koodi tunnisteet. Hitsauksen loputtua telemetriatieto lähetetään SOAP- tai REST-rajapintaan riippuen käytettävästä IoT-laitteesta. Kaikilta IoT-laitteilta lähtevä hitsaustelemetriä lähetetään XML-formaatissa.

Hitsaaja voi lähettää IoT-laitteeltaan myös käyttäjäviestejä. Käyttäjäviesteihin kuuluu muun muassa: Hitsauspalon valmistuminen, Hitsin valmistuminen, syykoodi ja monia muita. Riippuen lähettävästä laitteesta käyttäjäviestien rakenne ja formaatti voivat erota. Vanhempi Smart Reader lähettää tiedon XML-formaatissa, kuin taas uudempi X8 käyttää JSON-formaattia.

REST-toteutus vastaanottopalvelimelle on toteutettu Express-kirjaston päälle. Express mahdollistaa Node.js sovelluksen helpon ja nopean tavan toteuttaa REST-pohjainen toteutus.

Vastaanottopalvelimen arkkitehtuuri on pyritty pitämään mahdollisimman jatkokehitettävänä. REST- ja SOAP-rajapinnat on toteutettu omalla tavallaan. Rajapinnat tekevät tämän jälkeen REST-pohjaisia pyyntöjä AWS Lambda http-päätepisteisiin.

```
const AWS = require('aws-sdk');
const zlib = require('zlib');
const _ = require('lodash');

const Logger = require('./helpers/logger');
const log = new Logger();

const config = require('../config.json');
function upload(data, serialNumber, cb) {
    // If upload is enabled then push data to stream
    const uploadEnabled = _.get(config, 'UPLOAD_ENABLED', true);
    if (uploadEnabled) {
        compress(data, (err, zip) => {
            if (err) {
                cb(err);
            }
            const kinesis = new AWS.Kinesis({
                region: config.SERVERLESS_REGION
            });
            kinesis.putRecord({
                StreamName: config.KINESIS.STREAM_NAME,
                PartitionKey: serialNumber,
                Data: zip
            }, (err, data) => {
                if (err) {
                    return cb(err);
                }

                log.debug(data);
                cb(null, true);
            });
        });
    } else {
        cb(null, true)
    }
}

function compress(data, cb) {
    zlib.deflate(JSON.stringify(data), {
        level: zlib.constants.Z_BEST_COMPRESSION
    }, (err, buf) => {
        if (err) {
            cb(err);
        }
        cb(null, buf);
    })
}
```

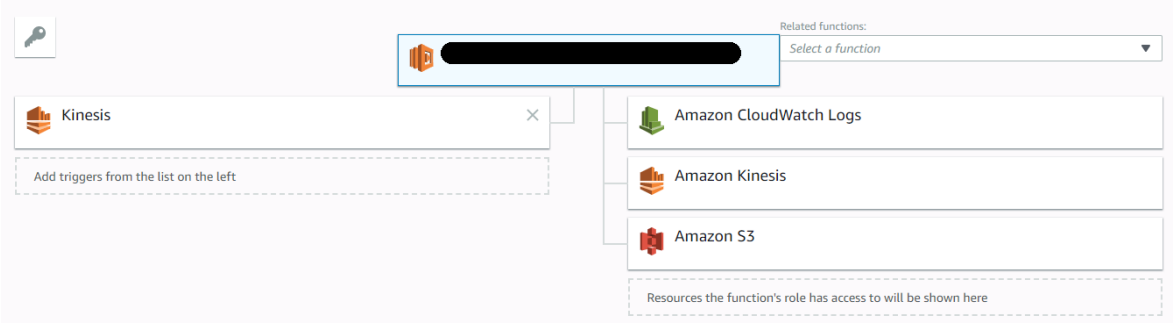
KUVIO 13. Telemetriatiedon lähetys Kinesis-virtaan

Kuviossa 13 esitetään telemetriatiedon lähetys Kinesis-virtaan.

Kinesis-virtaan lähetettävä tieto saa maksimissaan olla 1 MB kooltaan. Ennen lähetystä tieto pakataan. Tällöin telemetriatieto saadaan mahtumaan 2-5 kertaa pienempään tilaan.

## 4.5 Telemetriatiedonkäsittely

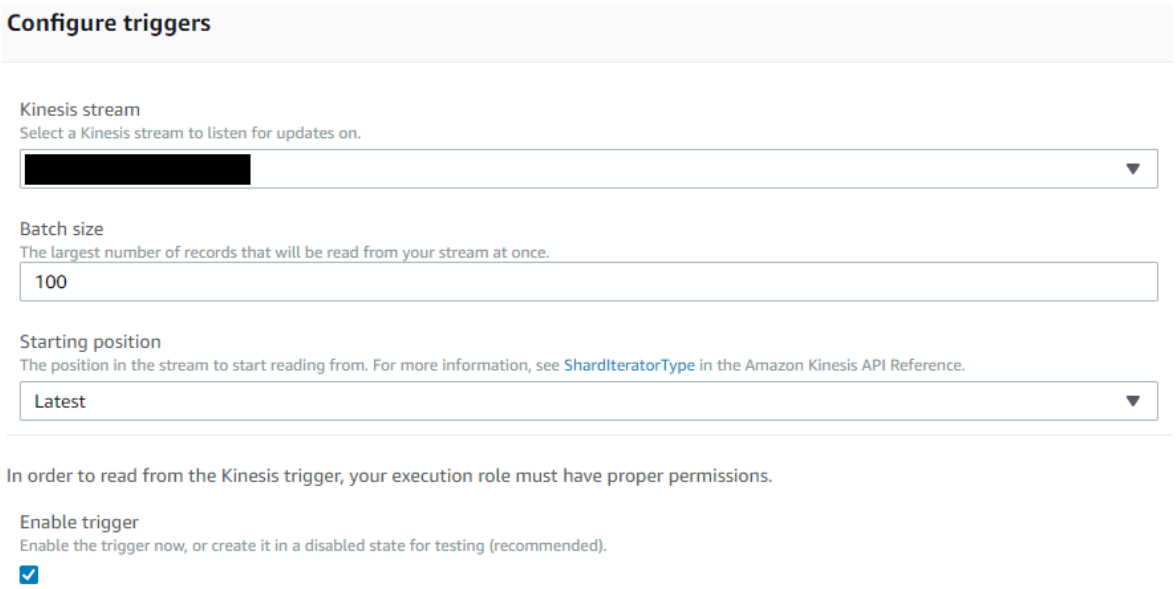
Telemetriatiedonkäsittely on toteutettu AWS Lambdalla. Lambda mahdollistaa funktion liittämisen Kinesis Data Stream -virtaan. Telemetriatiedonkäsittely sisältää kaksi eri lambdafunktiota. Toinen prosessoi kaiken tiedon, joka lähetetään Kinesis-virtaan, ja toinen varmuuskopioi kaiken tiedon. Koska lambda mahdollistaa funktion liittämisen Kinesis-virtaan, voidaan uusia ominaisuuksia jatkokehittää vain toteuttamalla uusi lambdafunktio ja kiinnittämällä se Kinesis-virtaan.




KUVIO 14. Telemetria lambdan resurssinäkymä

Kuviossa 14 näytetään lambdafunktion resurssinäkymää, jossa lambdafunktioon on kiinnitetty Kinesis-virta. Kehittäjä voi myös kiinnittää muita palveluita lambdafunktioon. Näihin kuuluu muun muassa S3-palvelu.

Kiinnittäessä lambdafunktion Kinesis-virtaan kehittäjä voi määrittää muutaman asetuksen. Nämä asetukset on esitetty kuviossa 15.




KUVIO 15. Kinesis-virran ja lambdafunktion asetukset

Ylimpänä voidaan valita luotu Kinesis-virta listasta. Valittu Kinesis-virta on se, mitä luetaan. ”Batch size” -asetus määrittää, kuinka monta tietuetta maksimissaan luetaan yhdessä lambdafunktion suorituksessa. Kinesis-virran sisältäessään enemmän tietueita, kuin maksimi eräkoko, suoritetaan lambdafunktio uudestaan, kunnes kaikki tietueet on luettu. ”Starting position”-asetus määrittää, mistä kohtaa Kinesis-virtaa tietueita aletaan lukemaan.

Telemetrian käsittelyssä eräkooksi on määritetty 20. Kyseiseen lukuun päädyttiin suorittamalla muutamia testejä eri pituisilla telemetriatiedostoilla ja kyseinen luku oli sopiva keskiarvo.

Lambdafunktion maksimi suoritusaika on viisi minuuttia ja maksimi muistinkäyttö 3008 Mb. Isompien eräkokojen prosessointi on nopeampaa, mutta prosessointi käyttää tällöin enemmän muistia. Prosessoidessa riittävän pieniä eräkokoja saadaan taas telemetriatietoa näkymään nopeammin käyttöliittymässä.

Telemetriatiedonkäsittelyn on oltava mahdollisimman suorituskykyinen johtuen siitä, että käsittelylle on asetettu tavoitteeksi maksimissaan 5-10 sekuntia. Eri toimintojen ajaminen rinnakkain, kuten varmuuskopiointi ja prosessointi, on yksi suorituskykyä lisäävä ominaisuus. Lyhyen välimuistin käyttö HTTP-pyynnöissä telemetriatiedon viitetiedostojen haussa on toinen.

### 4.5.1 Telemetria

Telemetriatiedonkäsittely hallitsee hitsauslaitteiden lähettämät hitsaustelemetriat. Näihin kuuluvat hitsauksen aikana tallennettu telemetriatieto ja käyttäjän lähettämät viestit hitsauslaitteelta.

Kuviossa 16 näytetään lambdafunktio, joka prosessoi Kinesis-virrasta luetut tietuet ja suorittaa oikean funktion tietueille.

```
export default async function telemetryAction(config, event) {
    let records: any[];
    try {
        config.log.debug({ event }, 'telemetryAction called');
        records = await unzip(event);
        config.log.debug({ records }, 'unzipped records');
        config.log.info(`Running telemetry on ${records.length} records`);
        // Welds
        let telemetryV1: any[] = [];
        // User events
        let telemetryV2: any[] = [];

        for (let record of records) {
            let recordData = _.get(record, 'kinesis.data');
            const recordEvent = checkEventType(recordData);
            switch (recordEvent.type) {
                case EventType.weldData:
                    telemetryV1.push(record)
                    break;
                case EventType.reasonCode:
                case EventType.passComplete:
                case EventType.weldComplete:
                    if (recordEvent.isARCQ) {
                        _.set(record, 'kinesis.data', await ArcQFormatter(config, recordData));
                    }
                case EventType.workOrderComplete:
                case EventType.workOrderInactive:
                case EventType.workOrderActive:
                    telemetryV2.push(record)
                    break;
                default:
                    config.log.warn('Unknown event provided')
                    break;
            }
        }
        // Handle telemetry V1 (welds)
        if (telemetryV1.length > 0) {
            config.log.debug({ telemetryV1 }, 'telemetryV1 records');
            await analyzer(config, telemetryV1);
        }
        // Handle telemetry V2 (user-events)
        if (telemetryV2.length > 0) {
            config.log.debug({ telemetryV2 }, 'telemetryV2 records');
            await userEvent(config, telemetryV2);
        }
        return true;
    } catch (error) {
        config.log.error('Fatal error happened while processing records, cannot continue');
        await errorHandler(error, records, config);
        return true;
    }
};
```

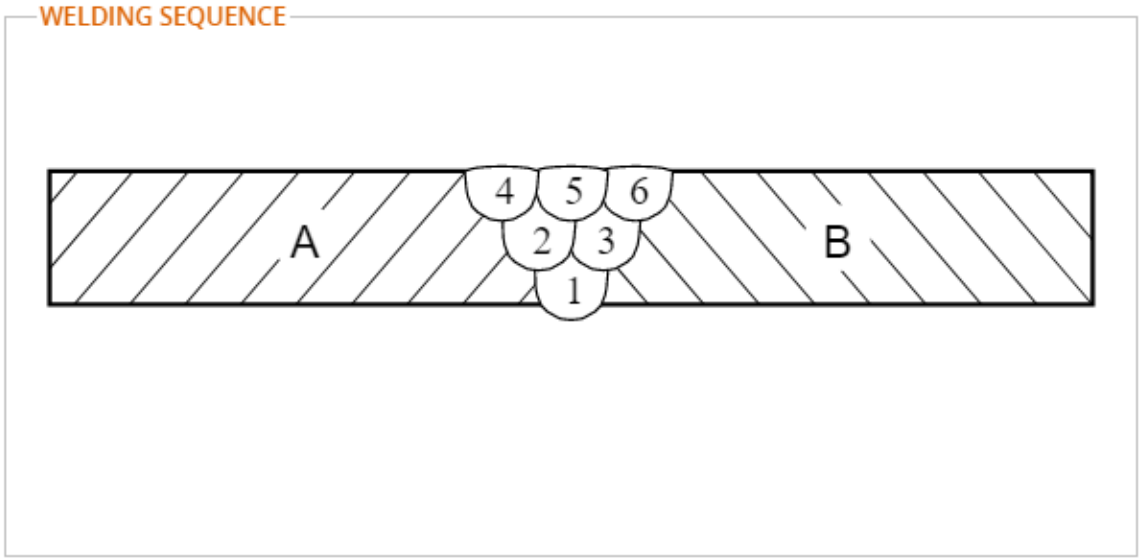
KUVIO 16. Hitsaustelemetrian ja käyttäjäviestien hallinta

Hitsauslaitteet lähettävät kahdenlaista tietoa, hitsaustelemetriaa ja käyttäjäviestejä.

Käyttäjäviesteillä tarkoitetaan muun muassa: hitsauspalon valmistuminen, hitsin valmistuminen ja syykoodi.

Riippuen lähettävästä laitteesta käyttäjäviestien rakenne ja formaatti ovat erilaisia. Uudempi X8 MIG Welder lähettää tiedon JSON-formaatissa, kuin taas Smart Reader XML-formaatissa.

Hitsauspalon valmistuminen ja hitsin valmistuminen ovat oleellisimmat käyttäjäviesteistä. Kyseiset käyttäjäviestit mahdollistavat telemetrianseurannan tietylle hitsille. Hitsaaja voi tehdä monta hitsauspalkoa yhdelle hitsille. Kuviossa 17 on määritetty, että osat A ja B hitsataan kuudella hitsauspalolla.




KUVIO 17. Hitsauspalot WeldEye Drawing tool -työkalussa

Jokaisen hitsauspalon hitsaamisen jälkeen hitsaaja lähettää hitsauspalko valmis -käyttäjäviestin. Käyttäjäviesti sisältää palon pituuden ja uniikin id:n, jolla viitataan hitsiin. Hitsauspalon valmistumisessa haetaan kaikki yksittäiset hitsausvedot hitsauspalon id:llä ja kyseisistä hitsausvedoista otetaan ylös mahdolliset viitetiedostot. Viitetiedostoihin kuuluu muun muassa hitsausohje, hitsari, lisäaine, lisänaineenerä. Hitsausvedoista lasketaan yhteenlaskettu kaariaika, keskiarvot hitsausjännitteestä ja virrasta. Hitsauspalon pituutta, kaariaikaa ja hitsausjännitettä ja virtaa käytetään laskemaan lämmöntuonti.

Mitä suurempi lämmöntuonti, sitä enemmän tuodaan lämpöenergiaa hitsiin ja vastaavasti sitä hitaammin hitsi jäähtyy. Jos teräs ei ole koostumukseltaan voimakkaasti karkeneva, käyttämällä suurta lämmöntuontia saatetaan estää karkeneminen muutosvyöhykkeessä tietyssä hitsauskohteessa. Liian suurella lämmöntuonnilla teräksen iskusitkeys heikkenee. (Ovako 2018, 6.)

Vetokokeessa pienillä kuormitusnopeuksilla huoneenlämpötilassa metallit yleensä murtuvat sitkeästi. Murtuminen voi olla hauras, kun kuormitetaan suuridimensioisia kappaleita

tai kun kuormitus tapahtuu suurella nopeudella (iskumainen kuormitus) alhaisessa lämpötilassa. Haurasmurtumistaipumusta korostavat kappaleessa esiintyvät lovet tai muut teräväkulmaiset viat. Haurasmurtumisen vastustuskykyä sanotaan iskusitkeydeksi. (Wikipedia 2018.)

Lämmöntuonnin laskenta auttaa huomattavasti hitsien laadunvalvonnassa. Kyseinen ominaisuus automatisoi lämmöntuonnin laskennan. Huomattaessa liian suuri lämmöntuonti tietylle hitsille, voidaan kohta korjata välittömästi. Kuviossa 18 on leikkaus WeldEye:n hitsilista näkymästä.

| | | WELDING PARAMETERS | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Arc time | Completed | WPS | Welder | Current | Voltage | Travel speed | Length | Heat input |
| 00:00:29 | ✓ | 1 | Jarmo Puro | 1023 A | 39.9 V | 533 mm/min | 258 mm | 3.67 kJ/mm |
| 00:00:57 | ✓ | 1 | Jarmo Puro | 253 A | 14.7 V | 1911 mm/min | 1816 mm | 0.09 kJ/mm |

KUVIO 18. Leikkaus hitsilistasta

$$\text{Hitsausenergia } E = \frac{\text{Hitsausvirta} \times \text{Kaarijännite} \times \text{kesto}}{\text{pituus}}$$

$$\text{Lämmöntuonti } Q = k \times E$$

KUVIO 19. Kaavat hitsausenergian ja lämmöntuonnin laskemiseen.

Täyttämällä arvot (1023 A * 39.9 V * 29 sekuntia) / 258mm kuvion 19 kaavaan voidaan laskea hitsausenergia. Laskemalla saadaan hitsausenergiaksi 4588 J/mm. MIG/MAG-hitsauksessa terminen hyötysuhde eli k on 0,8 (Ovako 2018, 6). Täyttämällä lämmöntuoton kaavan kuviossa 19 saamme 0.8*4588 J/mm. Lämmöntuotoksi tulee täten 3670.4 J/mm. Kilojouleiksi muutettuna lämmöntuonti on 3.67 kJ/mm arvoon. Tyypillinen lämmöntuonti MIG/MAG hitsauksessa on 0,5-3 kJ/mm (Ovako 2018). Kyseinen hitsi on tehty hitsaussimulaattorilla, joka ei salli hitsausparametrien muuttoa.

Hitsaustelemetriatiedon Smart Reader ja X8 MIG Welder ja mahdolliset muut laitteet lähettävät samassa formaatissa. Kyseinen formaatti on todettu hyväksi tavaksi tallentaa telemetriatietoa. Hitsaustelemetrian rakenne on esitetty kuviossa 20.

```
<file version="0.5" generated="2017-12-15T04:51:57">
  <generator serial="123456789" mode="KAS" version="1.04.00.0">QRex</generator>
  <welddata>
    <barcode timestamp="2017-12-15T04:04:22" type="welder">4343</barcode>
    <barcode timestamp="2017-12-15T04:04:29" type="wire">12.51</barcode>
    <barcode timestamp="2017-12-15T04:06:06" type="wps">135-07</barcode>
    <weld starttime="2017-12-15T04:51:57.573" id="00000001-0001-0f0c-07e1-0f0433392446">
      <machine memchipid="" wfboardid="PSNK1234567" psboardid="KA123456"
       productfamily="FastMig Synergic" productname="KMS"/>
      <sampleset timespan="welding">
        <sample timestamp="2017-12-15T04:51:57.573">
          <value parameter="weldingprocess">MIG/MAG</value>
          <value parameter="memorychannel">10</value>
          <value parameter="weldingprogramnumber">0</value>
          <value parameter="cablelength">0</value>
          <value parameter="wisepenetration">false</value>
          <value parameter="wisefusion">false</value>
          <value parameter="minilog">false</value>
          <value parameter="weldingvoltage">57.7</value>
          <value parameter="weldingcurrent">1</value>
          <value parameter="wirefeedspeed">3.4</value>
          <value parameter="wirefeedercurrent">0.1</value>
          <value parameter="weldingvoltagecompensated">57.7</value>
        </sample>
        <sample timestamp="2017-12-15T04:51:57.675">
          <value parameter="weldingvoltage">27.9</value>
          <value parameter="weldingcurrent">265</value>
          <value parameter="wirefeedspeed">9.9</value>
          <value parameter="weldingvoltagecompensated">27.9</value>
        </sample>
        <sample timestamp="2017-12-15T04:51:57.781">
          <value parameter="weldingvoltage">29.1</value>
          <value parameter="weldingcurrent">275</value>
          <value parameter="wirefeedspeed">10.1</value>
          <value parameter="wirefeedercurrent">0.3</value>
          <value parameter="weldingvoltagecompensated">29.1</value>
        </sample>
        <sample timestamp="2017-12-15T04:51:57.874">
          <value parameter="weldingvoltage">30.7</value>
          <value parameter="weldingcurrent">281</value>
          <value parameter="wirefeedercurrent">0.5</value>
          <value parameter="weldingvoltagecompensated">30.7</value>
        </sample>
      </sampleset>
      <summary preweldarctime="0" weldarctime="400" postweldarctime="0" arctime="400"/>
    </weld>
  </welddata>
</file>
```

KUVIO 20. Hitsauslaitteen lähettämä raaka XML-tiedosto

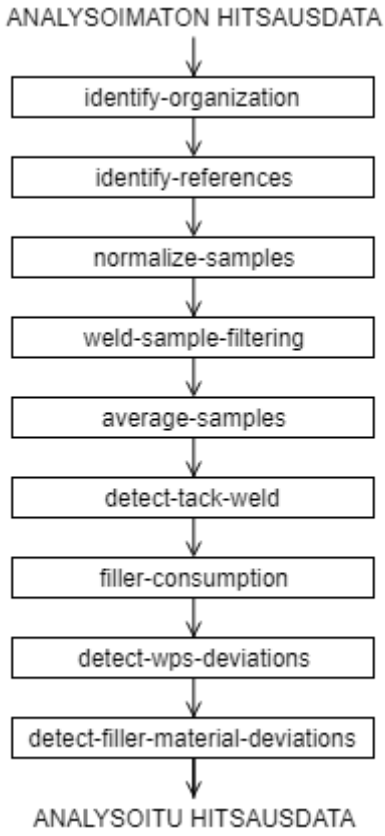
Hitsaustelemetrian analysointi vaatii, että hitsauslaitteen lähettämä XML-tiedosto konvertoidaan JSON-formaattiin. Tätä konvertointia varten on toteutettu ”WeldXmlToJson-niminen kääntäjä.

XML-kääntäjä oli alun perin toteutettu xml2js-NPM paketilla. Ensimmäisten versioiden jälkeen huomattiin, että noin neljän minuutin hitsaustelemetriatiedon konvertoinnissa kuluu noin 170 millisekuntia aikaa. XML-kääntäjä oli yksi pullonkaula koko toteutuksessa. Tutkimalla eri paketteja, jotka kääntävät XML-formaatista JSON-formaattiin päästiin lopputulokseen, että ”fast-xml-parser”-niminen NPM-paketti on nopein. Vaihtamalla NPM-paketista toiseen XML-tiedoston konvertoinnista saatiin noin 120 millisekuntia pois.

Yhdessä hitsaustelemetria tiedostossa voi olla useita tuhansia näytteitä riippuen hitsausajan pituudesta. Jokaisessa näytteessä on aikaleima, jokaisen näytteen aikaväli lasketaan millisekunnin tarkkuudella. Aikavälin laskemiseen käytettiin Moment.js NPM-kirjastoa.

Moment.js on kevyt JavaScript-kirjasto aikaleimojen jäsennykseen, validoitiin, manipulointiin ja muotoiluun (NPM 2018). Moment.js mahdollistaa helpon ja nopean aikaleimojen muokkaamisen. Samassa neljän minuutin hitsaustelemetriatiedostossa huomattiin, että aikaleiman laskennassa kului noin 70ms aikaa. Puhtaalla JavaScript-toteutuksella ilman ylimääräisiä kirjastoja samassa toteutuksessa kului noin 45ms. XML-tiedoston konvertointiin tehtiin myös muutamia pieniä muutoksia, jotka paransivat suorituskykyä vielä lisää. Lopullinen XML-konversio samalla neljän minuutin hitsaustelemetriatiedostolla kaikilla suorituskykyä parantavilla muutoksilla kestää noin 25ms.

Hitsaustelemetrien analysointi on toteutettu käyttäen putket ja suodattimet -arkkitehtuurimallia. Analysointi on toteutettu kehittämällä useita suodattimia, jotka ottavat sisään dokumentin, tekevät jonkun tietyn operaation, jossa muokkaavat dokumenttia ja antavat dokumentin eteenpäin mahdolliselle seuraavalla suodattimelle.




KUVIO 21. Analysointiprosessi

Ylimmät viisi suodatinta kuviossa 21 ovat esiprosessointi-suodattimia analysointiprosessissa. Nämä suodattimet pitää suorittaa tietyssä järjestyksessä ennen muita suodattimia.

**Identify-organization**

”Identify-organization”-suodattimella tunnistetaan mille organisaatiolle hitsaustelemetria kuuluu IoT-laitteen sarjanumeron perusteella. Laitteen tunnistamisen jälkeen haetaan tälle organisaatiolle kuuluvat asetukset. Asetukset tallennetaan analysoitavaan hitsaustelemetriadokumenttiin, jolloin muut suodattimet voivat käyttää samoja asetuksia ilman, että niiden tarvitsee asetuksien hakua uudelleen.

**Identify-references**

”Identify-references”-suodattimella tunnistetaan viittaukset. Viittauksilla tarkoitetaan dokumentteja, joihin viivakoodit viittaavat. Viittaukset ovat mitä hitsaaja on valinnut Smart Reader- tai X8 Control Pad -laitteellaan. Viittauksiin kuuluu muun muassa hitsaaja, hitsausohje, lisäaine, suojakaasu.

```
/**
 * Fetches Person by the person's id/barcode.
 */
async getPersonAsync(doc, config: WeldAnalyzerConfig, token) {
    const tenantId: string = _.get(doc, 'tenantId');
    const personId: string = _.get(doc, 'data.user.id', undefined);
    if (personId) {
        const url = `${config.API_URL}/pq/personnel/${encodeURIComponent(personId)}?type=barcode`;
        const person: any = await http(url, {
            method: 'GET',
            headers: {
                authorization: `Bearer ${token}`
            }
        }, `${tenantId}-person-${personId}`);
        if (person) {
            _.set(doc, 'references.personnel', person);
        }
    }
    return;
}
```

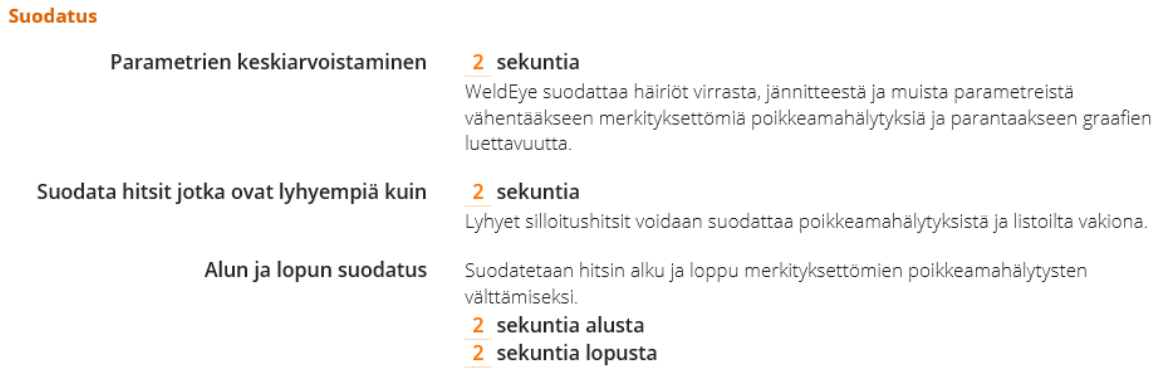
KUVIO 22. Hitsaajan viitedokumentin haku

### Normalize-samples

”Normalize-samples”-suodattimella normalisoidaan näytteet. Normalisoinnilla näytteet pilkotaan 100ms pituisiin näytteisiin ja näistä näytteistä lasketaan keskiarvot hitsausjännitteelle ja virralle.

### Weld-sample-filtering

”Weld-sample-filtering”-suodattimen tehtävä on tarkistaa jokainen näyte ja määrittää, onko näyte ”preweld”, ”welding” vai ”postweld” -vaihetta. Preweld ja postweld -asetuksilla voidaan määrittää, mitä aluetta ei prosessoida poikkeamien varalta. Asetukset ovat kuuma-aloitus ja kraatterintäyttöä varten. Kuuma-aloituksella hitsausteho on suurempi hitsauksen alussa, millä pyritään vähentämään huonon liitoksen mahdollisuutta. Kraatterintäytöllä hitsaushitsaustehoa vähennetään hitsauksen lopussa, että kraatteri voidaan täyttää.

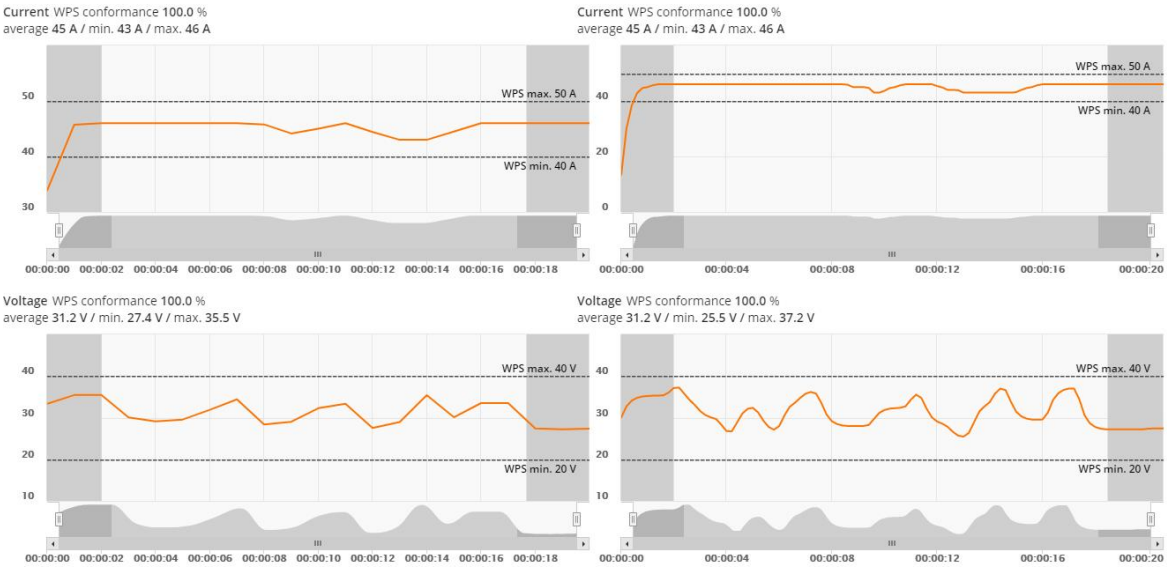


KUVIO 23. Suodatusasetukset WeldEye-käyttöliittymässä

**Average-samples**

”Average-samples”-suodatinta käytetään näytteiden keskiarvoistukseen. Keskiarvoistamisella poistetaan ylimääräiset piikit ja häiriöt näytteistä, jotka eivät vaikuta lopputuloksen laatuun. Tällä saavutetaan se, että todella lyhyet virta- tai jännitepiikit eivät aiheita poikkeamaa hitsaustelemetrian analysoinnissa.

Kuviossa 24 näkyy ero 0.2 sekunnin ja 2 sekunnin keskiarvoistuksen välillä. Kuviossa näkyy myös, jos hitsauksen alussa ei käytettäisi 2 sekunnin suodatusta, virta olisi aiheuttanut poikkeaman analysointiprosessissa. Suodatusalueet esitetään harmaalla graafin alussa ja lopussa.




KUVIO 24. Keskiarvoistetut hitsaustelemetriat WeldEye-käyttöliittymässä

**Detect-tack-weld**

”Detect-tack-weld”-suodatinta käytetään tunnistamaan, onko analysoitava hitsi silloitushitsi. Silloitushitsit ovat lyhyitä hitsejä, joilla pyritään pitämään hitsattavat kappaleet yhdessä hitsauksen aikana. Silloitushitseiksi määritetään kaikki hitsit, jotka ovat lyhyempiä kuin asetussivulla määritetty sekuntimäärä.

```
export default class DetectTackWeldFilter implements WeldAnalysisFilter {
    name = 'detect-tack-weld';

    async runAsync(doc: TelemetryV2Envelope, config: WeldAnalyzerConfig) {
        const start = new Date().getTime();

        let arcTime: number = _.get(doc, 'data.weld.arcTime', undefined);
        let tackTime: number = _.get(doc, 'analysed.weld.configuration.tackTimeMs', undefined);

        _.set(doc, 'analysed.weld.isTackWeld', false);

        if (tackTime && arcTime && _.isNumber(tackTime) && _.isNumber(arcTime)) {
            let isTackWeld: boolean = false;

            if (arcTime < tackTime) {
                isTackWeld = true;
            }
            _.set(doc, 'analysed.weld.isTackWeld', isTackWeld);
        }

        const end = new Date().getTime();
        setFilterDuration(doc, this.name, start, end);
    }
}
```

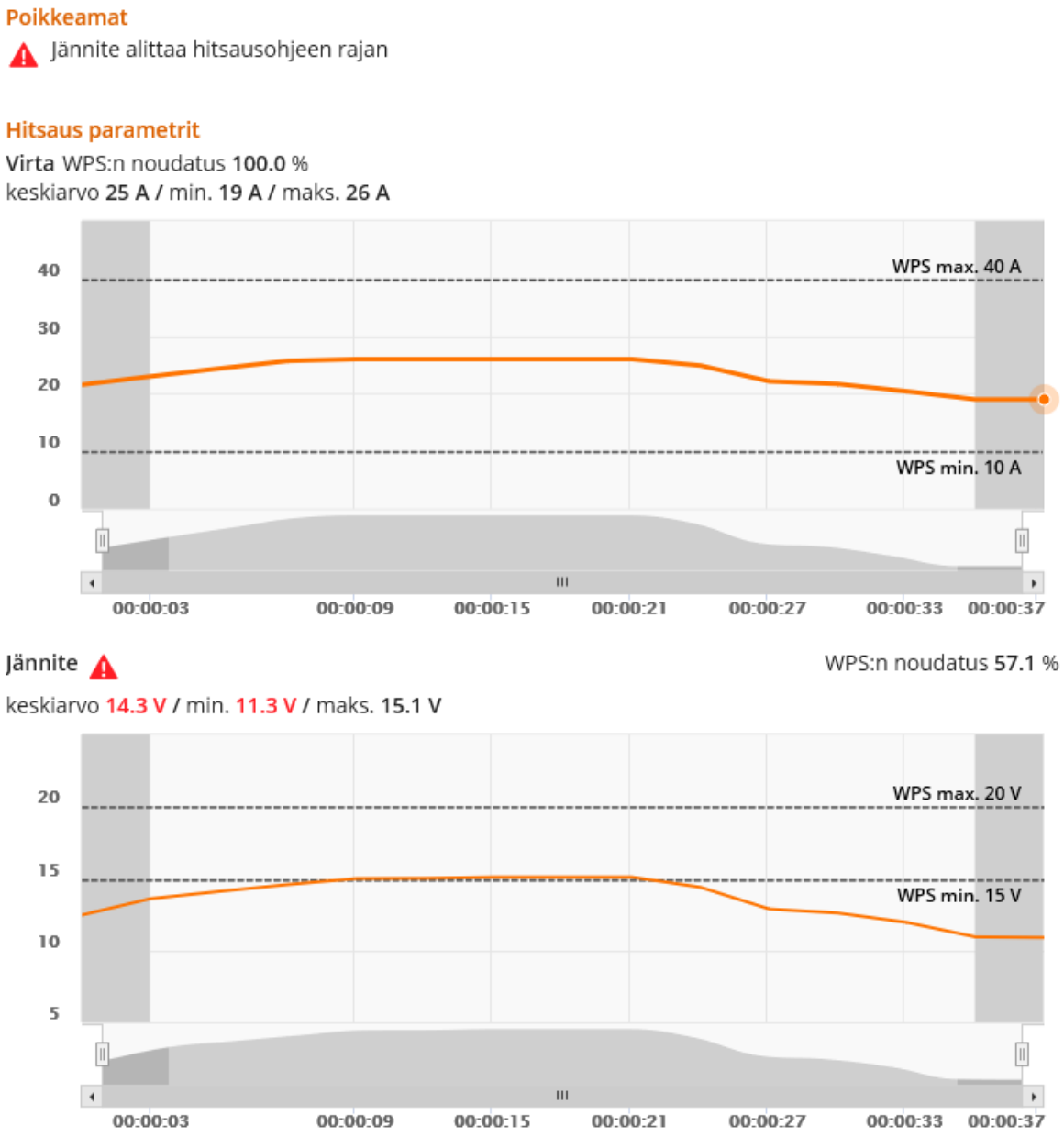
KUVIO 25. Silloitushitsien tunnistuksen lähdekoodi

**Filler-consumption**

”Filler-consumption”-suodatinta käytetään laskemaan mahdollisen lisäaineen kulutuksen metreissä ja grammoissa. Laskettua lisäaineenkulutusta voidaan tällöin verrata esimerkiksi ostettuun määrään ja seurata kulutusta ja hävikkiä.

**Detect-wps-deviations**

”Detect-wps-deviations”-suodatinta käytetään tunnistamaan hitsausohjeiden poikkeamat. Suodatin vertaa hitsaustelemetrian virta- ja jännitearvoja hitsausohjeen ylä- ja alarajoihin. Rajojen ylittäessä analysoituun telemetriatietoon merkitään, millainen poikkeama tapahtui. Mahdolliset poikkeamat voidaan tällöin näyttää käyttöliittymässä. Käyttäjä voi tällöin seurata hitsaustapahtumia ja mahdollisen poikkeaman tapauksessa tehdä vaadittavia toimenpiteitä.




KUVIO 26. Poikkeama hitsausjännitteessä

**Detect-filler-material-deviations**

”Detect-filler-material-deviations”-suodatin tarkistaa, onko lisäaine oikea. Asetuksista voidaan myös määrittää, tarkistetaanko lisäaineen paksuus. Väärän lisäaineen tapauksessa analysoituun hitsiin asetetaan poikkeama ja poikkeama voidaan täten näyttää käyttöliittymässä.

Putket ja suodattimet -arkkitehtuurimalli mahdollistaa uusien suodattimien lisäämisen helposti ilman, että se vaikuttaa muuhun analysointiprosessiin.

### 4.5.2 Varmuuskopiointi

Kaikki Kinesis Stream -virtaan saapuva tieto varmuuskopioidaan. Varmuuskopioitu tieto tallennetaan samassa muodossa, kuin se on lähetetty IoT-laitteelta. Tämä mahdollistaa telemetriatiedon uudelleenkäsittelyn mahdollisien virhetilanteiden jälkeen.

Varmuuskopiointiin käytetään S3-palvelua. S3-palvelu on toteutuksensa ansiosta äärimmäisen turvallinen palvelu varmuuskopioida käyttäjän tieto. Tallentaessa olemassa olevan tiedoston päälle S3 pitää hallussaan vanhemmat versiot. Tämä mahdollistaa jäljitettävyyden, jos lähetettävän tiedoston sisältämä tieto muuttuu eri version yhteydessä.

Telemetriatiedon prosessointi on toteutettu siten, että sama tieto voidaan prosessoida useita kertoja uudelleen, eikä tämä aiheuta tiedon monistusta. Tämä mahdollistaa, jos isommassa tietomäärän prosessoinnissa tapahtuu virhe eikä tiedetä tai välitetä missä kohtaa virhe tapahtui, voidaan kaikki tieto ajaa uudelleen ilman mitään jälkivaikutuksia.

### 4.5.3 Virhetilanteet

Virhetilanteessa virheen aiheuttaneelle telemetriatiedolle annetaan aikaleima ja tämä tiedosto tallennetaan ”bad-data” S3-kansioon. Virhetilanteen jälkeen lambdafunktio jatkaa toimintaansa normaalisti seuraavalla tiedostolla. Virhetilanteen aiheuttanut tiedosto voidaan myöhemmin hakea S3-kansiosta ja tästä tiedostosta voidaan selvittää, onko tiedostossa jotain vikaa vai koodissa, joka suorittaa telemetriatiedon käsittelyn.

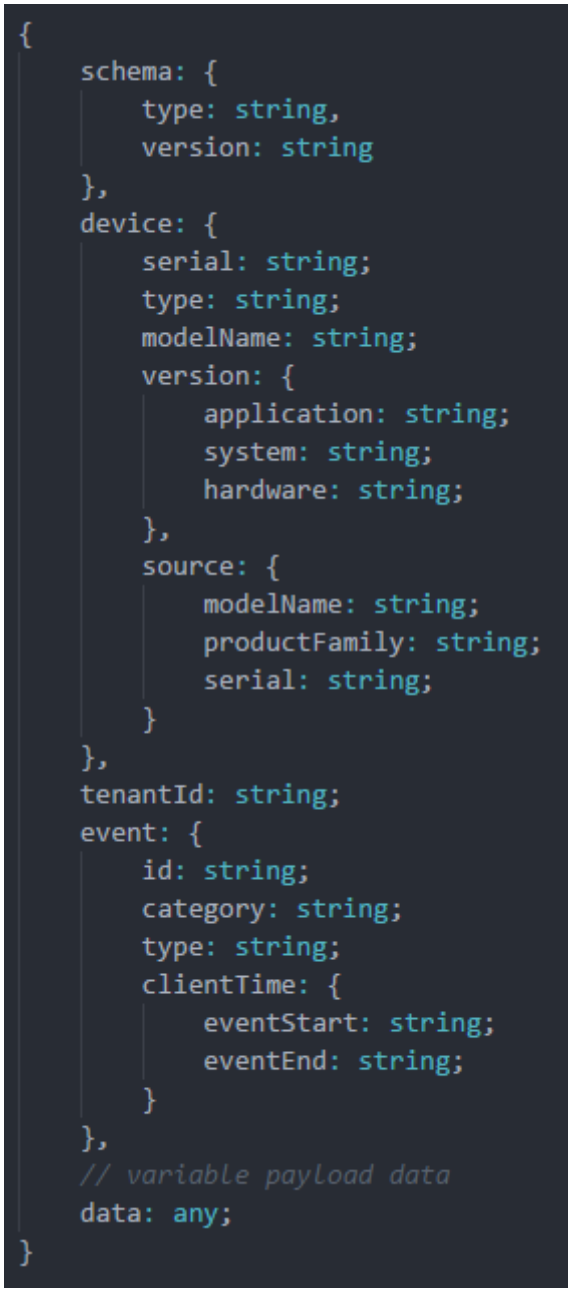
Virheen aiheuttanut tiedosto voidaan jälkeenpäin lähettää prosessoitavaksi uudelleen, jos huomataan, että telemetriatiedossa ei ole mitään vikaa. Tällöin virhe on voinut tapahtua esimerkiksi palvelinpuolen ohjelmistokoodissa, jokin HTTP-pyyntö on epäonnistunut. Vialinen telemetriatieto voi esimerkiksi tapahtua, jos hitsauskoneesta häviää sähköt kesken tiedoston kirjoituksen.

## 4.6 Tallennus

Jokainen telemetriatieto, jonka hitsauslaitteet lähettävät tallennetaan kahteen paikkaan. Ensimmäinen näistä on S3-pilvipalvelu. S3-pilvipalveluun tallennettu dokumentti sisältää kaiken tiedon, jonka analysointiprosessi tuottaa. Kyseistä dokumenttia käytetään, kun halutaan näyttää yksityiskohtaisempaa tietoa yksittäisestä telemetriatiedosta.

Toinen tallennuspaikka, johon hitsaustelemetria tallennetaan, on Elasticsearch. Elasticsearch mahdollistaa nopeat haku- ja aggregaatio-operaatiot hitsaustelemetrialla. Elasticsearch hakua käytetään esimerkiksi listaelementtien näytössä käyttöliittymässä. Dokumenttiin, joka on tallennettu Elasticsearch-indeksiin, on poistettu tiettyjä kenttiä, joita

ei tarvitse käyttää dokumenttien haku- tai aggregaatio-operaatioissa. Tämä vähentää Elasticsearch-indeksin käyttämää kiintolevytilaa ja nopeuttaa dokumenttien hakua.



```
{
    schema: {
        type: string,
        version: string
    },
    device: {
        serial: string;
        type: string;
        modelName: string;
        version: {
            application: string;
            system: string;
            hardware: string;
        },
        source: {
            modelName: string;
            productFamily: string;
            serial: string;
        }
    },
    tenantId: string;
    event: {
        id: string;
        category: string;
        type: string;
        clientTime: {
            eventStart: string;
            eventEnd: string;
        }
    },
    // variable payload data
    data: any;
}
```

KUVIO 27. Hitsaustelemetrian tietorakenne

Telemetriatieto ja käyttäjäviestit tallennetaan prosessoinnin jälkeen kuvion 27 mukaiseen tietorakenteeseen. Tietorakenteeseen kuuluu ”schema” eli skeema, joka kertoo minkä tyyppistä tietoa dokumentti pitää sisällään. Hitsaustelemetrian ja käyttäjäviestien tapauksessa skeeman tyyppi on ”TELEMETRY”.

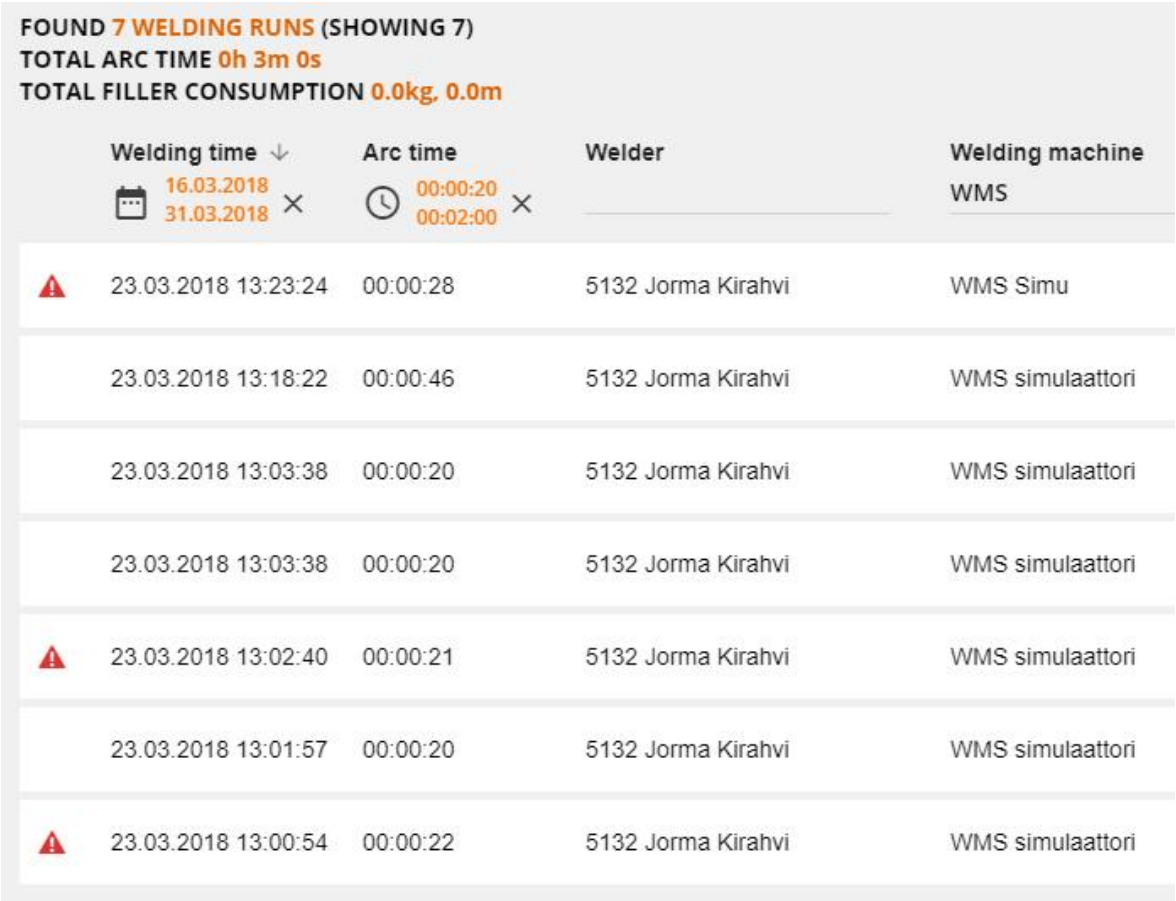
”Device”-osio pitää sisällään hitsauksen suorittaneen laitteen tietoja. Näihin tietoihin kuuluu muun muassa: sarjanumero, malli, ja laitteen versiotiedot.

Sovelluksen ollessa multi-tenant eli sovellus palvelee montaa eri asiakasta/organisaatiota, tallennetaan tietoon organisaation id. Tämä mahdollistaa sovelluksen toiminnan siten, että eri organisaation käyttäjät eivät voi hakea kenenkään muun tietoa.

”Event”-osiin tallennetaan tapahtuman uniikki id, tapahtuman luokka, tyyppi ja tapahtuman aikaleima laitteen kellosta.

## 4.7 Raportointi

Raportoinnissa hyödynnetään Hitsaustelemetrian Elasticsearch indeksiä. Elasticsearch mahdollistaa hitsien haun monella eri tapaa. Kuvio 25 on leikkaus WeldEye:n hitsausdata näkymässä, jossa on valittu aikaväli 16.3.2018 ja 31.3.2018 väli ja hitsien pituus 20 sekunnin ja 2 minuutin välillä, hitsauslaitteen nimessä pitää myös olla ”WMS”.



FOUND 7 WELDING RUNS (SHOWING 7)
TOTAL ARC TIME 0h 3m 0s
TOTAL FILLER CONSUMPTION 0.0kg, 0.0m

| | Welding time ↓ 16.03.2018 31.03.2018 × | Arc time 00:00:20 00:02:00 × | Welder | Welding machine WMS |
|---|---|---|---|---|
| ⚠ | 23.03.2018 13:23:24 | 00:00:28 | 5132 Jorma Kirahvi | WMS Simu |
| | 23.03.2018 13:18:22 | 00:00:46 | 5132 Jorma Kirahvi | WMS simulaattori |
| | 23.03.2018 13:03:38 | 00:00:20 | 5132 Jorma Kirahvi | WMS simulaattori |
| | 23.03.2018 13:03:38 | 00:00:20 | 5132 Jorma Kirahvi | WMS simulaattori |
| ⚠ | 23.03.2018 13:02:40 | 00:00:21 | 5132 Jorma Kirahvi | WMS simulaattori |
| | 23.03.2018 13:01:57 | 00:00:20 | 5132 Jorma Kirahvi | WMS simulaattori |
| ⚠ | 23.03.2018 13:00:54 | 00:00:22 | 5132 Jorma Kirahvi | WMS simulaattori |

KUVIO 28. Leikkaus WeldEye-käyttöliittymän hitsausdata näkymästä

Kuviossa 29 on esitetty JSON-objekti, joka sisältää kuvion 28 hakuparametrit. Hakuparametrit eivät ole täysin Elasticsearch dokumentaation mukaisia. Syy tähän johtuu tietokantaluokasta, joka on toteutettu väliin mahdollistaen tietokannan vaihdon ilman muitten toteutuksien muutosta.

```
{
    "from": 0,
    "size": 50,
    "sort": [
        {
            "time.eventStart": {
                "order": "desc"
            }
        }
    ],
    "parameters": [
        {
            "match": {
                "analysed.weld.isTackWeld": false
            }
        },
        {
            "range": {
                "time.eventStart": {
                    "gte": "2018-03-16T00:00:00.000Z"
                }
            }
        },
        {
            "range": {
                "time.eventEnd": {
                    "lte": "2018-03-31T23:59:59.999Z"
                }
            }
        },
        {
            "range": {
                "data.weld.arcTime": {
                    "gte": 20000,
                    "lte": 120000
                }
            }
        },
        {
            "match": {
                "references.weldingMachine.name": {
                    "query": "WMS",
                    "operator": "and"
                }
            }
        }
    ]
}
```

KUVIO 29. Hakuparametrit kuviossa 28 esiintyville hitseille

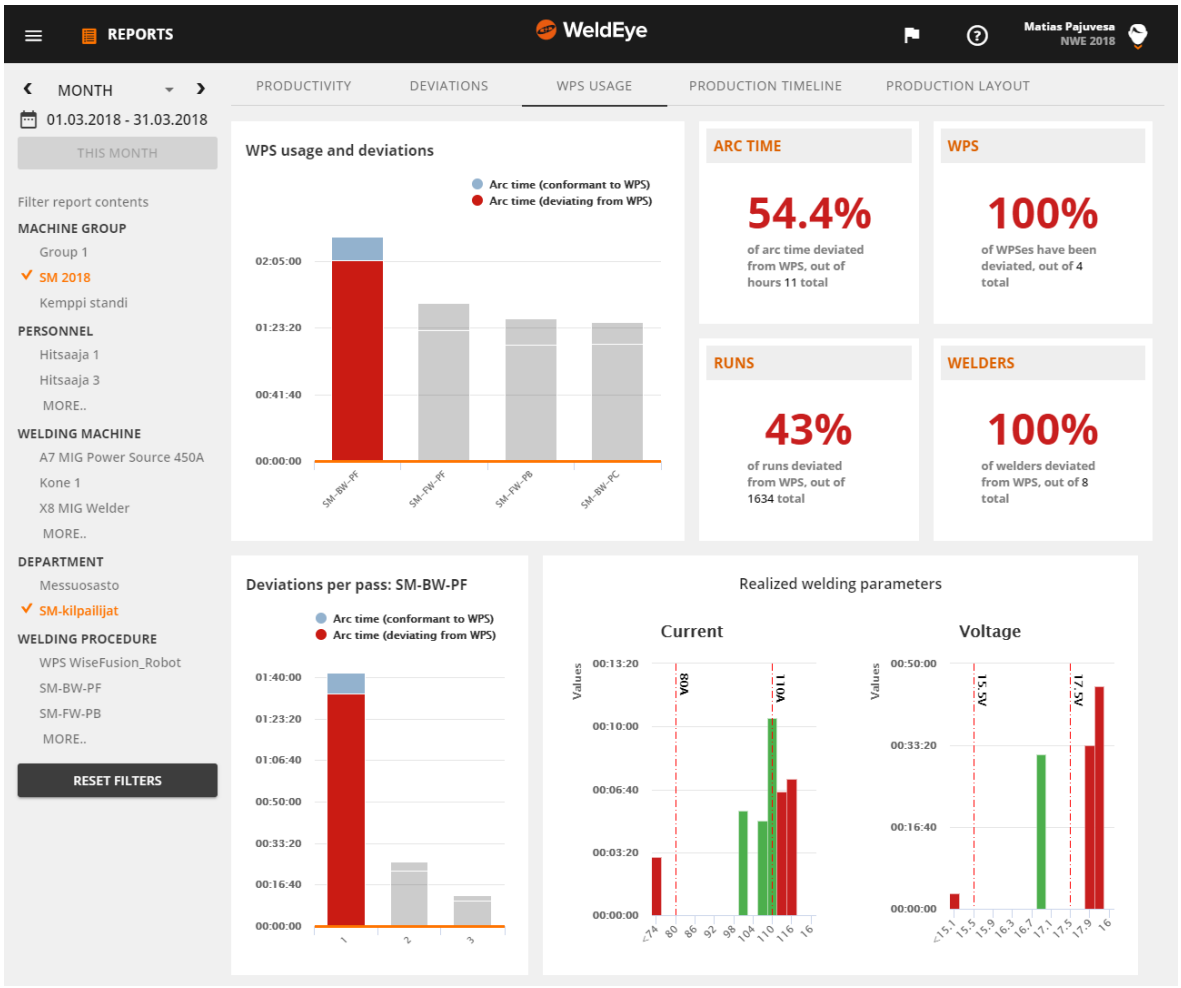
Hitsaustelemetrian hakuoperaatiot suoritetaan kuviossa 30 esitetyn lambdafunktion läpi. Lukuisat eri osat käyttöliittymässä ja monet muut lambdafunktiot käyttävät kyseistä lambdafunktiota. Palvelinpuolessa mikropalveluarkkitehtuuri käyttö mahdollistaa kyseisen toteutuksen. Lambdafunktioiden skaalautuminen automaattisesti poistaa kyseisen yksittäisen lambdafunktion hidastumisen ja kaatumisen.

```
export default async function searchAction(config, event) {
    config.log.debug('searchAction called', { 'event': event });
    config.iam.isAllowedTo('weldeye.qc.welddata.read');

    const body = _.get(event, 'body');
    if (!body) {
        config.log.info({ event }, 'Body missing');
        return null;
    }
    try {
        const storage = new ObjectStorage({ log: config.log, iam: config.iam, options: config.storageOptions });
        const response: any[] = await storage.queryRaw('es', body);
        return response;
    } catch (e) {
        throw new Errors.InternalServerError('Internal server error');
    }
};
```

KUVIO 30. Hitsaustelemetrian hakulambda

Elasticsearch mahdollistaa tiedon monimuotoiset haku- ja aggregaatio-operaatiot. Haku- ja aggregaatio-operaatiot auttavat WeldEye-pilvipalvelun raportit-näkymän kehityksessä. Aggregaatiot mahdollistavat muun muassa: minimien, maksimien, keskiarvojen, prosenttien ja monien muitten laskennan miljoonista dokumenteista. Nämä aggregaatiot avustavat monimuotoisten raportointiominaisuuksien kehittämisessä. Kuviossa 31 on esitetty raportointi näkymää WeldEye-pilvipalvelussa. Kyseinen näkymä näyttää raportointitietoa 17.-19.3.2018 järjestetyssä Nordic Welding Expo:ssa nuorten SM-hitsauskilpailun analytiikkatietoa.




KUVIO 31. Raportit näkymä WeldEye-käyttöliittymässä

Kuvion 31 oikeassa alareunassa, ”Current”- ja ”Voltage”-pylväskaavioista voidaan nähdä, että suurin osa poikkeamista hitsausohjeeseen verrattuna on ylivirtaa ja jännitettä. Tästä voidaan päätellä, että hitsauskoneen virta ja jännitearvot ovat liian korkeat, tai hitsausohje on väärä kyseiseen hitsaukseen.

# 5 YHTEENVETO

Työn tavoitteena oli toteuttaa toimiva prosessi hitsaustelemetrian käsittelylle jatkuvasti kasvavassa pilvipalvelussa. Hitsaustelemetrian käsittelyyn kuului telemetriatiedon vastaanottaminen, analysointi, tallentaminen ja analysoidun telemetriatiedon raportointi.

Prosessi toteutettiin Amazon Web Services (AWS) -pilvilaskenta-alustan päälle. Amazon Web Services tarjoaa laajan kokoelman erilaisia pilvipohjaisia palveluita, jotka tarjoavat erinomaiset mahdollisuudet kehittää pilvipohjaisia sovelluksia riippumatta sovelluksen tyypistä tai koosta.

Opinnäytetyön tuloksena on toimiva ratkaisu. Amazon Web Services -pilvilaskenta-alusta mahdollisti prosessin toteutuksen sulavasti. Mikropalvelinarkkitehtuuri antoi hyvät edellytykset kehittää skaalautuvaa ja helposti jatkokehitettävää sovellusta. WeldEye-pilvipalveluun kehitetään uusia ominaisuuksia joka viikko. Iso osa uusista ominaisuuksista käyttää hitsaustelemetriatietoa hyödykseen.

Serverless-arkkitehtuuri nopeutti sovelluksen kehitystä jättämällä pois infrastruktuurin hallinnan. Kehittäjänä pystyin keskittymään vain sovelluksen kehitykseen. Tämä jo itsestään nopeutti sovelluksen kehitystä huomattavasti.

Prosessin toteutuksessa suurimmat haasteet liittyivät sovelluksen skaalautuvuuteen, nopeuteen ja telemetriatiedon määrään. Hitsauslaitteet lähettävät huomattavan määrän telemetriatietoa ja kun tämä kerrotaan lukuisalla määrällä hitsauskoneita, telemetriatietoa on huomattava määrä. Luotettava ja skaalautuva prosessi mahdollistaa nopean ja vikasietoisen järjestelmän, jota on hyvä jatkokehittää tulevaisuudessa.

# LÄHTEET

AWS 2018a. Amazon EC2 Pricing [viitattu 19.3.2018]. Saatavissa: https://aws.amazon.com/ec2/pricing/

AWS 2018b. Amazon Elasticsearch Service [viitattu 26.2.2018]. Saatavissa: https://aws.amazon.com/elasticsearch-service/

AWS 2018c. Amazon S3 [viitattu 26.2.2018]. Saatavissa: https://aws.amazon.com/s3/

AWS 2018d. AWS Global Infrastructure [viitattu 19.3.2018]. Saatavissa: https://aws.amazon.com/about-aws/global-infrastructure/

AWS 2018e. AWS Lambda Pricing [viitattu 5.4.2018]. Saatavissa: https://aws.amazon.com/lambda/pricing/

AWS 2018f. Cloud Products [viitattu 19.3.2018]. Saatavissa: https://aws.amazon.com/products/

AWS 2018g. Customer Success Stories from all Industries [viitattu 17.3.2018]. Saatavissa: https://aws.amazon.com/

AWS 2018h. What are Microservices? [viitattu 26.3.2018]. Saatavissa: https://aws.amazon.com/microservices/

AWS 2018i. What is Amazon EC2 [viitattu 17.3.2018]. Saatavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

AWS 2018j. What is Amazon Kinesis Data Streams [viitattu 17.3.2018]. Saatavissa: https://docs.aws.amazon.com/streams/latest/dev/introduction.html

AWS 2018k. What is AWS Lambda [viitattu 17.3.2018]. Saatavissa: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

codeacademy 2018. What is REST? [viitattu 26.3.2018]. Saatavissa: https://www.codecademy.com/articles/what-is-rest

Dzone / Web Dev Zone 2018. An Introduction to Node.js (Part 1) [viitattu 5.4.2018]. saatavissa: https://dzone.com/articles/introduction-to-nodejs-3

elastic 2018a. Basic Concepts. Started [viitattu 26.2.2018]. Saatavissa: https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html

elastic 2018b. Elasticsearch [viitattu 26.2.2018]. Saatavissa: https://www.elastic.co/products/elasticsearch

Inspecta 2018. Hitsauksen laadunvarmistus (ISO 3834). [viitattu 5.4.2018]. Saatavuus: https://www.inspecta.fi/Palvelut/Sertifiointi-ja-arviointi/Johtamisjarjestelmasertifiointi/valmistava-teollisuus/Hitsauksen-laadunvarmistus-ISO-3834/

Kemppi Oy 2018a. KEMPPI – AND YOU KNOW [viitattu 13.3.2018]. Saatavissa: https://www.kemppi.com/fi-FI/

Kemppi Oy 2018b. Tietoja Kempistä [viitattu 15.3.2018]. Saatavissa: https://www.kemppi.com/fi-FI/yritys/kemppi/tietoa-yrityksesta/

Kemppi Oy 2018c. Welding production management [viitattu 24.3.2018]. Saatavissa: https://kemppi.studio.crasman.fi/pub/web/pdf/kemppi_welding-production-management_fi_FI.pdf

Kemppi Oy 2018d. X8 MIG Welder [viitattu 24.3.2018]. Saatavissa https://www.kemppi.com/fi-FI/tuotteet/tuote/x8-mig-welder/

Medium 2018. Serverless Computing [viitattu 19.3.2018]. Saatavissa https://medium.com/@MarutiTech/what-is-serverless-architecture-what-are-its-criticisms-and-drawbacks-928659f9899a

Node.js 2018a. About Node.js [viitattu 26.3.2018]. Saatavissa: https://nodejs.org/en/about/

Node.js 2018b. Node.js [viitattu 26.2.2018]. Saatavissa: https://nodejs.org/en/

NPM 2018. moment [viitattu 20.3.2018]. Saatavissa: https://www.npmjs.com/package/moment

Ovako 2018. Ovakon terästen hitsaus. [viitattu 28.3.2018]. Saatavuus: http://www.ovako.com/PageFiles/49/Ovakon_terasten_hitsaus_15724.pdf

StackExchange 2018. What makes an application scalable? [viitattu 26.3.2018]. Saatavissa: https://softwareengineering.stackexchange.com/questions/65080/what-makes-an-application-scalable

TechTarget 2018. REST (REpresentational State Transfer) [viitattu 17.3.2018]. Saatavissa: http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer

VirtuaaliAMK 2018. SOAP, Simple Object Access Protocol [viitattu 26.3.2018]. Saatavissa: http://www2.amk.fi/mater/tietotekniikka/soap/3_soap.html

W3C 2018. 1. Introduction [viitattu 17.3.2018]. Saatavissa: https://www.w3.org/TR/soap12-part1/

Weldeye. INSIGHT CREATES VALUE [viitattu 24.3.2018]. Saatavissa https://www.weldeye.com/en-US/

Wikipedia 2018. Haurasmurtuma [viitattu 5.4.2018]. Saatavissa: https://fi.wikipedia.org/wiki/Haurasmurtuma