TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Rasmus Luha

Andmekonveierite ehitamise ja paigaldamise automatiseerimine

Bakalaureusetöö (9 EAP)

Juhendaja:
Pelle Jakovits, PhD

Tartu 2025

Andmekonveierite ehitamise ja paigaldamise automatiseerimine

Lühikokkuvõte:

Kuigi praeguseks on andmekonveierite haldamiseks loodud mitemeid populaarseid tööriistu, nõuab uute konveierite ehitamine ja paigaldamine siiski endiselt omajagu aega ja vaeva. Selle töö eesmärk on uurida erinevaid andmekonveierite haldamiseks mõeldud tööriistu ning seejärel nende kasutamist lihtsustada ja automatiseerida. Töö lõpuks valmis tööriist mis on võimeline sisendi põhjal genereerima erinevaid andmekonveiereid töö jaoks välja valitud Nifi ja Telegrafi platvormidele.

Võtmesõnad: Andmekonveier, Nifi, Telegraf, Automatiseerimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Automating the building and deployment of data pipelines

Abstract:

Although by now there are many popular tools for managing data pipelines, the process of building and deploying them still takes a lot of time and effort. The goal of this thesis is to investigate different tools meant for managing data pipelines and to simplify and automate their usage. By the end of the thesis, a tool was created that can, from user input, generate different data pipelines for Nifi and Telegraf pipeline platforms, which where chosen for this thesis.

Keywords: Data pipeline, Nigi, Telegraf, Automating

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Kuulus fraas: “andmed on uus nafta”, on tõene mitmes tähenduses. Lisaks sellele, et mõlemad on tänapäevases majanduses väärtuslikud ressursid, sõltub ka andmete väärtus (just nagu nafta) sellest kui hästi seda on töödeldud e. rafineeritud. Ning just nagu nafta, nii on ka andmete liigutamiseks vaja konveiereid .[1]

Inglise keeles laialdaselt kasutatava termini *data pipeline* eestikeelseks vasteks on AKIT-is[1] (Andmekaitse ja infoturbe leksikonis) andmekonveier või andmetoru. Selles töös kasutatakse edaspidi terminit andmekonveier.

Andmete kiire kasv erinevates süsteemides, nagu IoT-seadmed, sotsiaalmeedia, pilveteenused, tehingutöötlussüsteemid jt. on tekitanud järjest suurema vajaduse kiirete ja tõhusate andmekonveierite järele [2].

Kuigi praeguseks on andmekonveierite haldamiseks loodud mitemeid populaarseid avatud lähtekoodiga tööriistu (nt Telegraf [3], Nifi [4], Dagster [5]), nõuab uute andmekonveierite ehitamine ja paigaldamine siiski endiselt omajagu aega ja vaeva. Lisaks tekib probleeme olukorras, kus on vaja luua suur kogus sarnaseid konveiereid ja iga konveier peab olema natuke erinev üksteisest - erinevate muutujate, ligipääsu info, konvertimisega. Kõiki neid konveieried ükshaaval käsitsi üles seadmine on tülikas.

Päriseluliste näitedena võib mõelda näiteks olukorra peale, kus mingi linn kogub andmeid erinevatest sensoritest üle linna ning soovib tuua kõik need andmed ühisesse andmebaasi. Oletame et andmete nimi ja tüüp jääb samaks ning seda ei pea andmekonveierite vaheliselt muutma. Sellises olukorras, kui sensorid kasutavad ka sama API-t, tuleb iga konveieri ehitamisel anda erinev API otspunkt, et see klapiks soovitud seonsoriga. Samuti võib tekkida olukord kus on soov erinevate sensorite andmed salvestada erinevatesse andmebaasidesse. Sellisel juhul tuleb iga andmekonveieri puhul, mis muidu võivad olla väga sarnased, muuta ära andmete laadimise lõpp-punkt. Olukorras kui linnal on näiteks 50 sensorit, või isegi rohkem, võib andmekonveierite ehitamine muutuda üsna tülikaks.
Sellises olukorras oleks väga mugav, kui konveierite ehitamist saaks automatiseerida. Näiteks selliselt, et oletades et ülejäänud konveieriosa jääb samaks, aga muuta on vaja vaid API otspunkt,

[1] https://akit.cyber.ee/

siis saaks anda mingile tööriistale, või skriptile mis seda tööriista kasutab, ette API otspunktide loendi, ning selle abil genereeritakse kõik 50+ andmekonveierit, erinevate otspunktidega.

Selle töö eesmärk on võrrelda erinevaid andmekonveierite haldamiseks mõeldud tööriistu. Seejärel uurida, kas ja kuidas saaks uute andmekonveierite loomis- ja haldamisprotsessi juures võimalikult palju vähendada käsitsi tehtava arendustöö mahtu, seda automatiseerides. Seejärel on eesmärk luua lahendus, mis võimaldab andmekonveierite genereerimist, aidates seeläbi kaasa konveierite ehitamise automatiseerimise ja lihtsustamise protsessile.

Esmalt on plaan taustapeatükis kirjutada admekonveieritest üldiselt, mis need on ning milliseid andmekonveierite tüüpe eksisteerib. Seejärel tuua selles peatükis välja ka eelevalt tehtud tööd, mis kattuvad antud töö teemaga ning miks antud töö on endiselt siiski vajalik. Samuti valida välja andmekonveierite ehitamise platvormid, mida võrrelda ja mida hiljem praktilises lahenduses kasutada, ning tutvustada neid taustapeatükis. Praktilise lähenemiskäigu esimeseks sammuks on vajalike tehnoloogiate ja tööriistadega tutvumine. Seejärel on plaan realiseerida praktlise töö peatükis lahendus ja kirjeldada seda ning selle loomisel tekkinud probleeme ja teadmust. Kui praktiline lahendus on valmis, siis on eesmärk lahendust testimise peatükis ka valideerida ning kirjutada seal testimiselt saadud tagasisidest ja sellest tehtud järeldustest.

2. Taust

See peatükk alustab ülevaatega sellest, mis on andmekonveierid. Seejärel uuritakse, kas ja kuidas on eelnevalt proovitud selle töö raames püstitatud probleemi lahendada. Lisaks tutvustatakse ka tehnoloogiaid, mis praktilse töö realiseerimiseks valiti, ning miks valiti just need.

2.1 Andmekonveierid

Gupta [6] defnieerib andmekonveiereid, abstraktselt võttes, kui jada erinevaid tööülesandeid, mis koguvad, töötlevad ja liigutavad andmeid erinevate süsteemide ja allikate vahel, vajades minimaalselt manuaalset vahelesegamist. Organisatsioonid tekitavad aina uusi andmeid ning töötlevad neid, et saadavast teadmusest kasu lõigata. Samas, andmete käitlemine on muutunud järjest keerulisemaks, vastavalt sellele mida rohkem on andmeid ja nende allikaid, ning andmekonveierid on muutunud *de facto* viisiks kuidas nende väljakutsetega tegeleda, kuna need pakuvad vähendatud keerukust/võimalust seda protsessi abstraheerida, samuti viisi andmete ja kogu protsessi läbivmonitoorimiseks ja jälgimiseks.

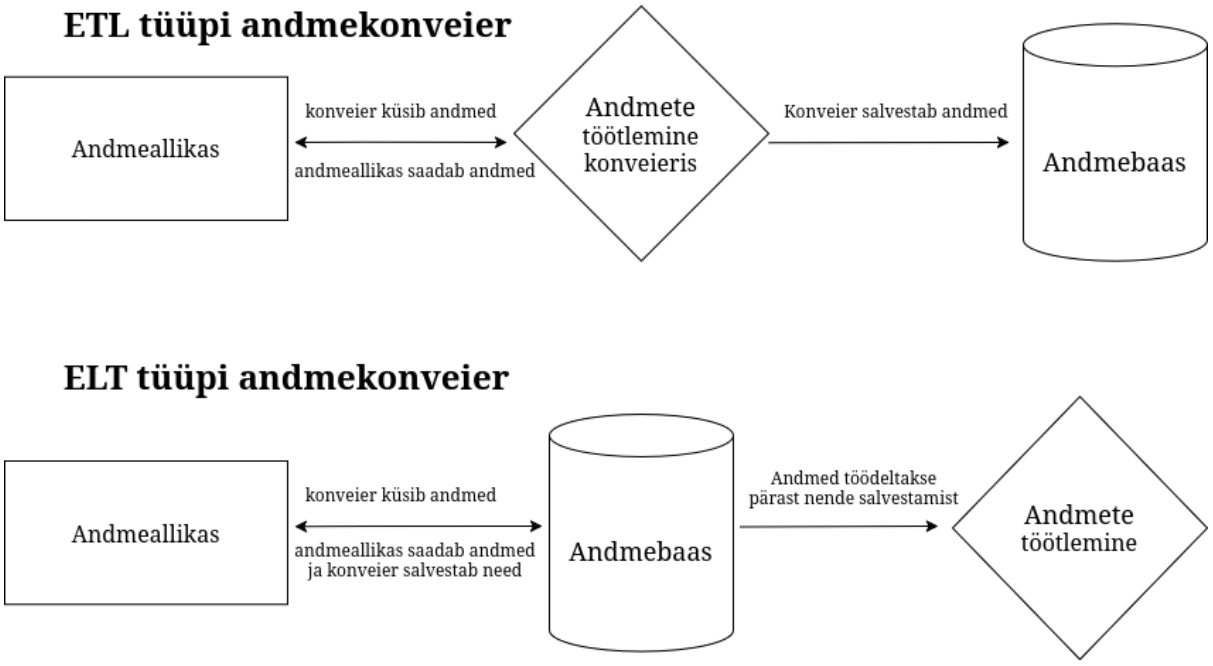
2.1.1 Andmekonveierite tüübid

Mbata jt. [7] töö põhjal saab andmekonveierite tüübid laias laastus jagada kolmeks. ETL/ELT tüüpi andmekonveierid, Masinõppe andmekonveierid ning Orkestreerimise ja töövoo haldamise (ingl *Workflow orchestration*) konveierid.

ETL/ELT andmekonveierid, mis on disainitud teostama erinevate süsteemide vahelist integratsiooni. ETL/ELT andmekonveieri nimi tuleb konveierie sammude inglisekeelsete nimede esitähtedest: *Extract* (ee ekstraktima, välja võtma), *transform* (ee transformeerima, muutma), *load* (ee üles laadima).
Andmed ekstaktitakse valitud allikast, filtreeritakse ja/või modifitseeritakse ning laetakse üles valitud sihtkohta. Võttes näiteks ühe ilmaandmete andmekonveieri mis tahab saada konkreetse linna temperatuuri, siis esimese sammuna ekstakteeritakse andmed, ehk siis tehakse ilmaandmete API-le päring ning saadakse vastu ilmaanded. Teise sammuna, transformeerimise all, otsitakse andmete hulgast välja soovitud linna andmed ja sealt omakord filtreeritakse ainult temperatuur. Teised võimalikud ilmaanded, näiteks sademete arv, tuulekiirus, päikesepaiste jne jäetakse tähelepanuta. Seejärel kolmanda ja viimase sammuna võeatakse välja filtreeridud soovitud linna temperatuuri andmed ning laetakse need andmekonveieris spetsifiseertud andmebaasi ülesse. Ning pärast määratud aja möödumist tehakse sama protsess uuesti e. jooksutatakse andmekonveierit uuesti ja saadakse uus hetketemperatuur.

Selliselt kasutaksegi andmekonveieried andmete hankimiseks, kogumiseks ja modifitseerimiseks, vajades selleks minimaalselt inimsekkumist.

Vastavalt vajadusele võib konveieri sammude järjekorda ka muuta, laadides andmed kõigepealt sihtkohta ning teostades andmetega soovitud muudatusi viimase sammuna (vt joonis 1). Erinevate sammude järjekorda võib käidelda ka kui erinevaid tüüpe aga selle töö raames on need gruppeeritud ühise ETL/ELT tüübi alla.



Joonis 1. ETL ja ELT tüüpi andmekonveierid teostavad samu samme, aga erinevs järjekorras

.

Masinõppe andmekonveierid on disainitud automatiseerima kogu masinõppe elutsüklit, alates andmete sisestamisest ja eeltöötlusest kuni featuuride töötlemise (ingl *feature engineering*), mudelite treenimise ja evitamiseni. Populaarsed tööriistad seda tüüpi konveierite jaoks on näiteks TensorFlow Extended (TFX)[2], Kubeflow[3], Metaflow[4].

Orkestreerimise ja töövoo haldamise konveierid tagavad omavahel seotud protsesside nõuetekohase ja korrapärase läbiviimise, automatiseerides seega andmete liigutamist ja töötlemist.

[2] https://www.tensorflow.org/tfx
[3] https://www.kubeflow.org/
[4] https:/metaflow.org/

Need konveierid ei tegele ise otseselt andmete töötlemise või liigutamisega, vaid haldavad protsesse, mis kasutavad teisi tööriistu. Näitena võib tuua näiteks Apache Airflow [5].

Antud töö käsitleb vaid ETL/ELT (edaspidi ETL) konveiereid, seda nii teoreetilises pooles, kus võrreldakse andmekonveierite ehitamise platvorme kui ka praktilises lahenduses, mis luuakse vaid seda tüüpi andmekonveieritele. Samuti võeti vastu otsus et andmekonveierite genereerimise tööriista ehitamisel fikseeritakse viimase sammu e. üleslaadimise lõpp-punkt. See tähendab et valitakse üks kindel andmebaas, mida lõpp-punktina kasutatakse. Seda tehakse selleks et piirata töö praktilise osa keerukust ja mahtu.

2.2 Eelnevad tööd ja lahendused

Varasemalt on tehtud töid, nt Donca jt [8] poolt, kus on proovitud kirjutada konveierite genereerimise tööriista. Loodi skript, mis jälgib Giti repositooriumi seisu, ning iga muutuse korral loob konveieri jaoks vajaliku *.yaml* faili, millest GitLab[6] oskab ehitada CI\CD konveieri (ingl continuous integration, continuous delivery; ee pidevintegratsioon, pidevvalmidus). Selle lahendusega genereeritavad konveierid polnud aga mitte andmekonveierid, vaid tarkvara tarnekonveierid (ingl *CI\CD pipeline*). Antud töö praktilise lahenduse eesmärk on teha sisuliselt sama, v.a mõned muutujad, nagu näiteks see, et repositoorimi jälgmise asemel küsitakse kasutaja sisendit ning ka see, et konveierid pole mitte CI/CD töövoo tarnekonveierid, vaid ETL tüüpi andmekonveierid.

Valdas [9] on teinud oma töös „ML-TOSCA: ML pipeline modelling and orchestration using TOSCA“ üsna sarnast asja selle töö eesmärgile. Ta ehitas oma töös erinevaid andmekonveierite komponente, mille omavahel integreerimise abil on võimalik lihtsutada ja automatiseerida konveierite ehitamise protsessi. Aga konveierid mis selle töö aluseks olid, polnud mitte selle töö teemaks olevad ETL andmekonveierid, vaid Masinõppe andmekonveierid.

Chilukoori jt. [10] on pakunud oma töös „Automation in Data Engineering: Challenges and Opportunities in Building Smart Pipelines“, et tulevikus võiks erinevate andmekonveierite haldamist automatiseerida tehisaru abil. Nad pakuvad, et kasutades tehisaru saaks sisse tulevaid andmeid automaatselt analüüsida, filtreerida ning transformeerida. Samuti võiks tehisharu abil monitoorida ning optimeerida konveieri ressursikasutust ja ennetada erinevaid probleeme,

[5] https://airflow.apache.org/
[6] https://about.gitlab.com/

mis võivad konveieri töö ajal tekkida. Antud töö eesmärgiks aga pole automatiseerimiseks kasutada tehisharu. Samuti erineb see, mida automatiseeritakse. Eelnevlat pakutud lahendus automatiseeriks pigem konveierite töövoogu, mitte aga nende ehitamist ning konfigureerimist.

Kõik kolm suurimat pilveplatvormi pakkujat [11]: Google, Amazon ja Microsoft pakuvad tööriistu, mis on mõeldud ETL tüüpi andmekonveierite mugavamaks loomiseks ja kasutamiseks. Google pakub tööriista Data Fusion [12], millel on *Drag and Drop* (ee lohista ja paigalda) tüüpi kasutajaliides, ning mille abil saab ilma koodi kirjutamata luua ETL konveiereid, kasutades vaid kasutajalt sisendina saadud *metadata*'t(ee metaandmed).
Microsofti Databricks [13] on mõeldud eelkõige suuremat hulka andmeid liigutavate konveierite jaoks ning erinvalt Google Data Fusionist puudub seal graafiline liides (*Drag and Drop*), ning metaandmed ja muu vajalik info kirjutatakse tööriista Pythoni *notebook*'i.
Amazon pakub tööriista Step Functions [14], aga see on mõeldud teistele AWS tööriistadele (AWS Glue, Lambda, S3) töövoo orkestreeriseks, kus kasutades erinevaid Amazoni pilveplatvormi tööriistu on ja neid *Step Functions* abil kokku tuues on võimalik luua erinevaid konveiereid.
Kõigi nende tööriistade puhul kehtib aga asjaolu, et nende kasutamiseks on vaja olla vastava pilveplatvormi klient ja need pole vabavaralised tööriistad.

Andmekonveierite ehitamise automatiseerimise võimalusi ja lahendust on uuritud ka näiteks Omogbai Oleghe jt. töös [15] „A framework for designing data pipelines for manufacturing systems", kus pakutakse välja raamistik, mille põhjal saaks luua andmekonveierite ehitamise automatiseerimise lahendusi. Välja pakutud raamistik kooseb viiest võtmekihist: andmeallikad, andmete liigutamine, andmete modifitseerimine, andmete salvestamine ning visualiseerimine. Selline raamistik on mõeldud standardiseerima andmekonveierite ehitamist, aidates seeläbi kaasa ehitamise automatiseerimisele. Töös pakuti antud raamistik küll välja, aga praktikas seda ei realiseeritud.
Antud töö praktilises osas on Oleghe jt. töös pakutud raamistik heaks verstapostiks, mille raamistiku saab võtta andmekonveierite genereerimise aluseks, kuid kus võib siiski teha muutusi vastavalt vajadusele.

2.3 Tehnoloogiate tutvustus

Töö raames valiti vabavaraliste ETL andmekonveierite ehitamise platvormide seast välja kaks, et kasutada neid võrdlemiseks ja praktilises lahenduse teostamiseks: Apache Nifi [4](edaspidi Nifi) ja InfluxData Telegraf [3] (edaspidi Telegraf). Nifi ja Telegraaf valiti, lisaks sellele et

mõlemad on vabavaralised, kuna neil on omavahel mitmeid erinevusi. Nifi kasutamiseks on brauseri põhine graafiline liides, kus toimub nii andmekonveierite ehitamine, paigaldamine kui ka tagasiside saamine ning monitoorimine. Telegraf aga on serveriagent, mille andmekonveierite ehitamine käib läbi konfiguratsiooni faili kirjutamise, ja selle siis paigaldatud Telegrafi agendile ette andmise.

Samuti on nende platvormide andmeonveierite masinloetav kuju erinevas formaadis. Nifi puhul JSON kujul, Telegrafi puhul aga TOML[7] kujul. TOML on minimalistlik konfigureermiskeel, kus süntaksi prioriteediks on selle lugemise mugavus.

Võib ka öelda, et need platvormid on ehitatud erinevaid kasutusjuhte silmas pidades. Telegrafi arendamisel on pigem arvestatud, et see pannakse kuhugi serverisse või virtuaalmasinasse taustal jooksma ning andmeid koguma, et need siis monitoorimiseks haldajale tagasi saata. Nifi on pigem aga mõeldud suure hulga erinevate andmekonveierite mugavaks jooksutamiseks ning haldamiseks. Neist kahest on Nifi kindlasti kompleksem ja rohkemate võimalustega tööriist. Järgnevalt tutvustatakse Telegrafi ja Nifit lähemalt, samuti tutvustatakse ka teisi praktilise lahenduse realiseerimise jaoks valitud tehnoloogiaid.

2.3.1 Apache Nifi

Nifi [4] on vabavaraline tööriist millel on veebipõhine kasutajaliides ja mis on mõeldud erinevate süsteemide andmevoogude haldamiseks.

Nifi kasutab konveierite loomisel protsessoreid [16], kus iga protsessor on disaintud teostama spetsiifilist alamülesannet kogu konveieri töövoost ning kombineeriedes erinevaid protsessoreid, saab paindlikult luua ning muuta erinevaid andmekonveiereid.

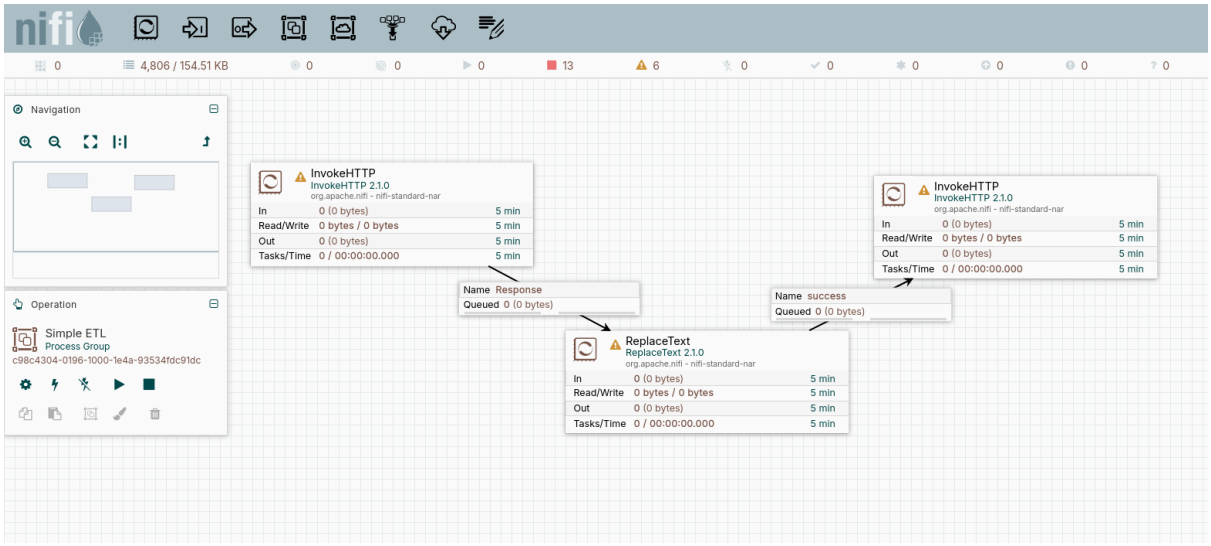
Näitena saab tuua ETL andmekonveieri loomise järgmiste protsessoritega:

InvokeHTTP protsessor läbi REST teenuse API-st andmete sisselugemiseks. ReplaceText protsessor, mis filtreerib sealt soovitud andmeväljad ning muudab nende kuju selliset, et järgmises sammus oleks võimalik andmed andmebaasi laadida. Ning viimase sammuna uuesti InvokeHTTP protsessor, mis võtab eelmise protsessori poolt saadetud filtreeritud andmed ning laeb need läbi REST teenuse etteantud andmebaasi. Joonisel 2 on selline andmekonveier ka välja toodud.

Nifil on ka oma ”*Expression Language*”[17], mille abil on võimalik konfigureerida protsessorite erinevaid väärtuseid dünaamiliselt ja seeläbi muuta konveiereid paindlikumaks. Kuigi Nifi konveierite ehitamine ja haldamine on mõeldud läbi veebiliidese, võimaldab platvorm ka

[7] https://toml.io/en/

konveierite (mille kohta öeldakse Nifi kontekstis ”*Processor group*”) importimist ja eksportimist JSON formaadis (vt joonis 3). See on väga kasulik antud töö kontekstis, sest JSON formaati faile on skripti abil tunduvalt lihtsam modifitseerida, kui kirjutada skript, mis teeb erinevaid API päringuid Nifi platvormi pihta, et sel viisil andmekonveier (*Processor Group*) luua.



Joonis 2. Näide minimalistlikust ETL andmekonveierist Nifi platvormil

```
..."processors": [
    {
      "identifier": "d7963d52-e8a3-3b27-9543-20db6cc99f6f",
      "instanceIdentifier": "9f1ceb8b-f048-3b47-5aec-773690811078",
      "name": "ReplaceText",
      "comments": "",
      "position": {
        "x": -208.0,
        "y": -968.0
      },
      "type": "org.apache.nifi.processors.standard.ReplaceText",
      "bundle": {
        "group": "org.apache.nifi",
        "artifact": "nifi-standard-nar",
        "version": "2.1.0"
      },
      "properties": {
        "Regular Expression": "(?s)(^.*$)",
        "Replacement Value": "placeholder",
        "Evaluation Mode": "Entire text",
        "Text to Prepend": null,
        "Line-by-Line Evaluation Mode": "All",
        "Character Set": "UTF-8",
        "Maximum Buffer Size": "1 MB",
        "Replacement Strategy": "Regex Replace",
      },
      "propertyDescriptors": {...
```

Joonis 3. Lõik Nifi andmekonveierist JSON formaati viiduna

2.3.2 Telegraf

Telegraf [3], mis on arendatud osana InfulxData platvormist [18], on vabavaraline tööriist, mis on mõeldud andmete kogumiseks, töötlemiseks ja edasi liigutamiseks valitud süsteemidesse. Telegraf paigaldatakse agendina soovitud süsteemi ning talle antakse ette konfiguratsioonifail (vt joonis 4), kus on defineeritud soovitud andmekonveier. Konfiguratsioonifailid on TOML-formaadis. Telegraf kasutab selleks, et defineerida konveieri ehitamiseks vajalike alamülessandeid, erinevaid pistikprogramme (ingl *pluggins*) [19].

```
[agent]
  debug = true
  interval = "3600s"


#INPUT: Laen andmeid Delta API-ist
[[inputs.http]]
  name_override = "telegraafi_deltaEnergy"
  urls = ["https://delta.iot.cs.ut.ee/measurement/measurements?source=780&dateFrom=2025-02-19

  method = "GET"

  # Autentimine
  username = "rasmus.luha"
  password = "TODO"

  # Vastuse formaat
  data_format = "json"

  # Täpsustan JSON-i välja
  json_query = "measurements"

  # Täpsustan alamvälja
  json_string_fields = ["KogEN.T.value"]

  # Ajatemplite konfiguratsioon - vajalikud andmebaasi jaoks
  json_time_key = "time"
  json_time_format = "2006-01-02T15:04:05Z"


# OUTPUT: Andmete laadmine andmebaasi
[[outputs.influxdb]]
  urls = ["http://influxdb:8086"]
  database = "telegraf_deltaEnergy"
  username = "admin"
  password = "admin"
```

Joonis 4. Näide Telegrafi konfiguratsioonifailist ETL konveierile

Nifiga võrreldes erineb Telegraf selle poolest, et puudub graafiline kasutajaliides ning konveierid on juba eos masinloetaval kujul. Üldse on Telegrafil vähem võimalusi, erinevad pistkprogramme/protsessoreid ning toetust eri tüüpi andmeallikatele. Selle võrra aga, nõuab Telegraf ka vähem ressursse. Telegraf ongi eelkõige mõeldud jooksma erinevates süsteemides taustal, kogudes meetrikat ja logisid. do

2.3.3 InfluxDB

Nagu andmekonveiereid tutvustava peatüki 2.1 lõpus kirjutati, siis otsutati, et valitakse välja üks kindel andmebaas andmekonveierite viimase sammu, ehk andmete salvestamise jaoks. Andmebaas, mis selleks valiti oli InfluxDB.

InfluxDB arendajate sõnul [20] on InfluxDB andmebaas, mis on loodud sündmuste ja ajaseeriaandmete (ingl *timeseries data*) kogumiseks, töötlemiseks, teisendamiseks ja salvestamiseks - ning see sobib suurepäraselt kasutusjuhtudeks, kus on vaja reaalajas andmete vastuvõttu ja kiireid päringuvastuseid, näiteks kasutajaliideste loomisel, monitooringusüsteemides ja automaatikalahendustes.

Neid omadusi, ning ka seda et InfluxDB on juba praegu kasutuses Delta maja automaatika süsteemi andmete hoidmiseks, arvesse võttes oletati, et InfluxDB sobib hästi IoT tüüpi andmete jaoks ja seega ka antud praktilise töö testandmete jaoks.

Kuna andmebaas on tööriistal fikseeritud, siis on ka fikseeritud millisel kujul andmeid tuleb andmebaasi saata. InfluxDB kasutab ajaseeriaandmete saatmiseks protokolli ILP (InfluxDB Line Protocol) [21], mille abil saab edastamiseks vajalikud andmed panna kirja ühe reaga selliselt:

```
measurementNimi võti1="väärtus1",võti2="väärtus2" ajatempel.
```

Siin on influxDB jaoks *measurement* justkui konteiner kus andmeid hoida ja mille abil neid organiseerida. Andmebaasi sees on *measuremendid measuremendide* sees on andmed.

Lisada toodud ILP protokolli näitele saab lisada ka näiteks silte (ingl *tags*), aga antud töö raames neid ei kasutata. Joonisel 5 on toodud näide InfluxDB kasutamisest.

2.3.4 Docker

Kõikide eelnevalt mainitud, töö jaoks vajalike komponentide (influxDB andmebaas, Telegraf agent, Nifi platvorm) jooksutatmiseks kasutati Dockeri[8] konteinereid. Konteineriseerimistarkvara, nagu Docker, võimaldab jooksutada erinevaid rakendusi eraldatud keskkonnas (konteineris) mugavdades seeläbi arendus- ja testimisprotsesse. See võimaldab kiiresti käivitada ja peatada soovitud arenduskomponente, tagades et keskkonnas kus need jooksevad pole mittesoovitud muutusi või lisategureid. Joonisel 5 on toodud näide InfluxDB Dockeri konteineris käivitamisest ning seejärel konteinerisse sisenemisest ja seal jooksva andmebaasiga suhtlemisest.

[8] https://docs.docker.com/get-started/docker-overview/

```
luhamus@muhalus:~/BAKA$ docker run -d --name test_influxdb -e INFLUXDB_DB=test_database -e INFLUXDB_ADMIN_USER=admin -e INFLUX
DB_ADMIN_PASSWORD=admin -p 8086:8086 influxdb:1.8
49f3b43aebbc38d972dd1d10d37f3a83d5fceccbe152799498281ccf83e9c8d1
luhamus@muhalus:~/BAKA$ docker exec -it test_influxdb influx

Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> show databases;
name: databases
name
----
test_database
_internal
> use test_database
Using database test_database
> INSERT temperature,sensor=abc value=23.5
> show measurements
name: measurements
name
----
temperature
> select * from temperature;
name: temperature
time                sensor value
----                ------ -----
1747146744769345774 abc    23.5
>
```

Joonis 5. Dockeri konteineri jooksutamine ja sellga suhtlemine InfluxDB näitel

Töö praktilise lahenduse realiseerimise jooksul kasutati koodi kirjutamise jaoks kahte programmeerimiskeelt, Bashi[9] ja Pythonit[10].

Järgmine peatükk katab praktilise lahenduse realiseerimise, selle käigus tehtud otsused ja samuti selle tulemusena valminud lahenduse.

[9] https://www.gnu.org/software/bash/manual/bash.html

[10] https://www.python.org/

3. Praktilise lahenduse realiseerimise

Praktilise lahenduse realiseerimise alguses pandi kirja kolm automatiseerimise taset, millest iga järgnev tase pakub suuremat paindlikust. Esimene tase: juba valmis ehitatud andmekonveierite, näiteks JSON või TOML faili konfiguratsioonifaili sisemiste parameetrite muutmine. Andmete allikat ega andmete lõpp-punkti, ehk andmebaasi muuta ei saa. Võimalik on aga muuta näiteks kui tihti andmekonveier jookseb ja milliseid API poolt tagastatavaid andmeid filtreerida.
Teine tase: sisendi (see mis andmeid tõmbab/kuulab) komponendi abstrahheerimine. Võrreldes eelmise tasemega pakub suuremat paindlikust sest andmeallikas pole enam fikseeritud mingile kindlale API-le, vaid on kasutaja poolt valitav.
Kolmas tase: sisend- ja väljundkomponentide valimine mitme võimaluse seast. See tase suurendab veelgi paindlikust, võimaldades lisaks esimese taseme võimalustele valida ka nii andmete sisendallika, kui ka lõpp-punkti. Kolmas tase annab kasutajale täieliku kontrolli kogu ETL andmekonveieri ehitamise üle. Kohe töö algusest tehti aga otsus, et viimase, kolmanda taseme teostus on raskusastmelt üle selle töö raamide ning läheks liiga ajakulukaks ja keerukaks. Seega pandi eesmärk selle töö raames saavutada teisel tasemel automatiseeritus.

Praktilise töö käigus kasutati lahenduse koodi kirjutamisel ChatGPT (2025 Aprill ja Mai GPT-4o ning GPT-4o mini mudel) abi probleemide lahendamisel ja konseptsioonide selgitamisel.

3.1 Andmed

Selleks, et praktilist osa realiseerida oli vaja testandmeid. Lahenduse arenduse alguses otsustati, et projekti arendamise raames valitakse kahed API poolt tagastatavad andmed. Esimene API mis valiti, oli Open-Meteo poolt pakutav tasuta ilmajaama API [22], mis valiti lihtsamateks katsetusteks, kuna selle tagastatavate andmete JSON kuju on üsna lihtnakoeline ja ilma suurema keerukuseta. Teine API, mis valiti, oli Cumulocity platvormil olev API Tartu Ülikooli Delta hoone päikesepaneelide andmetega. Selle API poolt tagastatavad andmed olid keerukamad ja erinevaid parameetreid oli rohkem. Näiteks oli seal võimalik valida kuupäevade vahemik, mille jooksul tekkinud andmeid küsida. Lisaks sai API kutses täpsustada lehe suurust (ingl *pageSize*), füüsilise instrumendi ID-d ja muud.
Need API-d valiti ka sellelepärast, et esimene neist, open-metro API, oli ilma autentimiseta ja kõigile kättesaadav. Delta andmete API vajas aga autentimist, kasutajanime ja parooli, iga API kutse tegemisel. Seega, kasutades neid API-sid saab praktilises lahenduses mõlemad olukorrad läbi katsetada ja lahendada. Mõlema API vastusest on toodud näited Lisa 4 all.

3.2 Lahenduse nõuded

Projekti alguses pandi paika funktsionaalsed ja mittefunktsionaalsed nõuded, mida lahenduse loomise käigus järgida ning meeles pidada. Funktsionaalsed nõuded kirjeldavad süsteemi konkreetseid tegevusi ja käitumist. Mittefunktsionaalsed nõuded kirjeldavad süsteemi soovitud omadusi ja kvaliteedi aspekte. Nõuete paika panemisel toetuti eelnevalt tehtud sarnaste tööde taustale ning samuti teadmusele, mis saadi Nifi ja Telegrafi platvormide peal andmekonveierite ehitamisest ja jooksutamisest ning andmete salvestamisest InfluxDB andmebaasi.

Funktsionaalsed nõuded

Järgnevalt on toodud paika pandud funktsionaalsed nõuded.

1. Lahendus suudab andmekonveierite konfigureerimise kohta sisendi saamisel genereerida soovitud töötava andmekonveieri.
2. Võimaldab valida toetatud platvormide vahel (selle töö raames Telegraf või NiFi), millele andmekonveier genereerida.
3. Kontrollib etteantud muutujate korrektsust ning probleemide korral tagastab veateate, mis selgitab, milles viga seisneb.
4. Probleemide puudumisel annab kasutajale tagasiside edukast andmekonveieri loomisest ja selle asukohast.
5. Võimaldab kasutajal kasutada tööriista interaktiivselt, ehk parsida algallika API vastuse andmeid jooksvalt, valides sealt andmekonveieri jaoks filtreeritavad andmed ning andes ka teised vajalikud sisendid tööriista jooksmise käigus.
6. Võimaldab kasutada tööriista mitteinteraktiivselt, ehk sisestada kõik vajalikud väärtused läbi konfiguratsioonifaili (ja mitte parsida api vastust).
7. Toetab kasutajalt järgmiste muutujate sisendi saamist:
    - sisendit kas soovitakse kasutada interraktiivset või mitteinterraktiivset kasutusmoodi
    - andmekonveieri nimi
    - algallika API url
    - vajadusel ka API kasutajanimi ja parool
    - algallika API vastusest filtreeritavad andmed

- andmekonveieri ajatsükkel (kui tihti andmekonveier jookseb)
- platvormi asukoha URL
- platvormi ligipääsuks vajalikud kasutajanimi ja parool.

Lisaks ka platvormipõhiseid muutujad, mida vastavalt platvormi valikule kas on või ei ole vaja.

8. Toetab influxDB andmbaasi kasutamist lõppsihtkohana ja selleks järgmiste vajalike sisendite saamist:
    - andmebaasi asukoha URL
    - andmebaasi nimi
    - andmebaasi kasutajanimi ja parool.
9. Kui valitakse Nifi platvorm, peab tööriist võimaldama konveieri sinna paigaldamise võimalust.
10. Lahendusel peab olema dokumentatsioon, mis selgitab kuidas tööriista kasutada ja konfigureerida, ning mis võimalused kasutajal on.
11. Projekti kood peab olema struktureeritud modulaarselt ning võimaldama hiljem lisada uusi mooduleid/andmekonveierite ehitamisplatvorme.
12. Lahendus peab sisendallika API-na toetama nii autetimist vajavaid kui ka mittevajavaid otspunkte.

Mittefunktsionaalsed nõuded

Järgnevalt on toodud paika pandud mittefunktsionaalsed nõuded.

1. Lahendus peab olema võimeline kasutama andmeallikatena erinevaid REST protokolli kasutavaid API-sid, mis tagastavad JSON kujul vastuse.
2. Lahendust peab saama mugavalt kasutada skriptimises ja olema deterministlik — seda saab kasutada automaatsetes töövoogudes selliselt, et pärast tööriista käivitamist jookseb see ilma kasutajapoolse sisendita.
3. Kui programmi jooksmise käigus tekib probleem, peab kasutajale tagastama veateated, mis selgitab probleemi ning kuidas seda on võimalik lahendada.

3.3 Lahenduse lähenemiskäigu kirjeldus

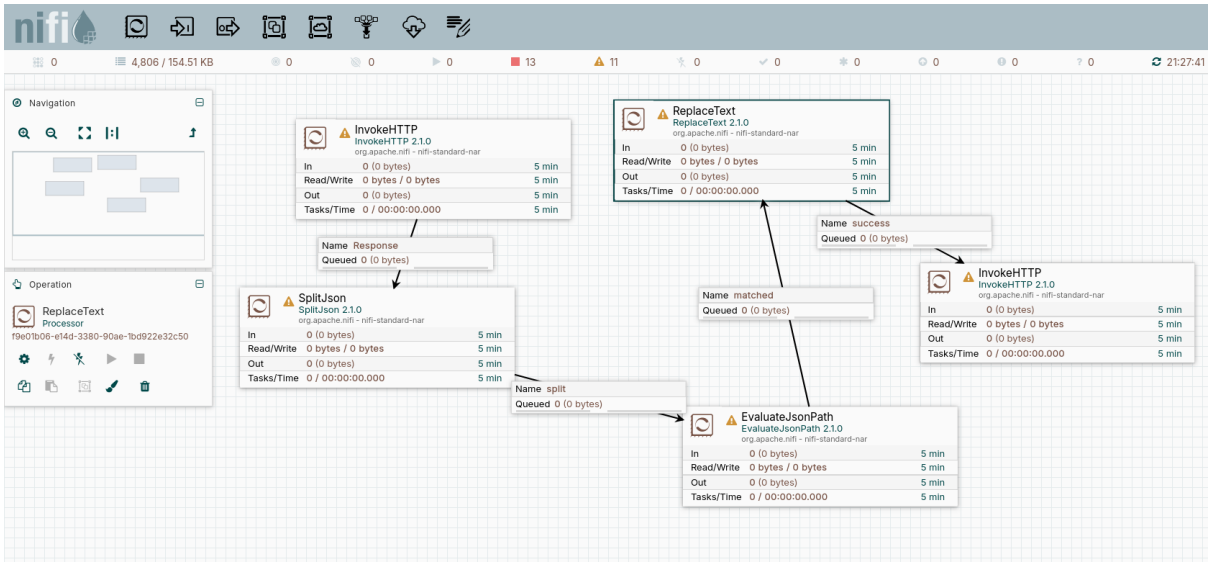
Lahenduse realiseerimisele läheneti plaaniga alustada lihtsamatest katsetustest, et teema ning sellest johtuvad probleemid arusaadavamaks teha. Seejärel oli eesmärk esmalt saavutada võimalikult kiiresti ülaltoodud tasemetest kõigepealt esimese taseme kondikava või prototüüp, panemata liigset aega detailide viimistlemisele. Seejärel võta sealt saadud teadmus ja info ning luua projekt, mis on võimeline realiseerima ka teisel tasemel automatiseerimist. Praktilise lahenduse loomiseks, pärast algseid võrdlusi ja katsetusi, valiti esimesena Nifi platvorm, kuna võrreldes Telegrafiga oli seal rohkem nüansse. Lisaks arvestati ka sellega, et eesmärk on kirjutada võimalikult palju funktsionaalsust selliselt, et seda oleks hiljem võimalik taaskasutada ka Telegrafi (või ka mõne kolmanda) platvormi projekti kaasamisel.

3.3.1 Esimesed katsetused ja prototüübid

Praktilise lahenduse esimesed katsetused algasid keskkonna püsti seadmisega. See hõlmas esmalt andmekonveierite ehitamisplatvormide Nifi ja Telegrafi paigaldamist. Samuti oli vaja paigaldada välja valitud InfluxDB andmebaas. Kõiki mainitud komponente jooksutati lokaalhosti konteinerites, kasutades selleks konteineriseerimistarkvara Docker.

Pärast seda, kui kõik vajalikud komponendid olid paigaldatud, oli järgmine samm luua mõlema andmekonveieri ehitamisplatovmile sama kasutusmalli (vt Lisa 1) põhjal andmekonveier, võrdlemaks platvormide erinevusi ning samuti selleks, et platvorme kasutama õppida.

Andmekonveier, mis loodi kasutas Delta päikesepaneelide API-t, tõmbas sealt andmed ning filtreeris andmetest päikesepaneelide kilovatt-tundide väärtuse. Seejärel valmists see andmete kuju ette selliselt , et InfluxDB oskaks neid vastu võtta ning salvestas siis andmed andmebaasi. Loodud andmekonveierite kujud on välja toodud joonistel 6 ja 7.



Joonis 6. Lisa 1 kasutusmalli alusel loodud konveier Nifi platvormil

```
[agent]
  debug = true
  interval = "3600s"


#INPUT: Laen andmeid Delta API-ist
[[inputs.http]]
  name_override = "telegraafi_deltaEnergy"
  urls = ["https://delta.iot.cs.ut.ee/measurement/measurements?source=780&dateFrom=2025-02-19

  method = "GET"

  # Autentimine
  username = "rasmus.luha"
  password = "TODO"

  # Vastuse formaat
  data_format = "json"

  # Täpsustan JSON-i välja
  json_query = "measurements"

  # Täpsustan alamvälja
  json_string_fields = ["KogEN.T.value"]

  # Ajatemplite konfiguratsioon - vajalikud andmebaasi jaoks
  json_time_key = "time"
  json_time_format = "2006-01-02T15:04:05Z"


# OUTPUT: Andmete laadmine andmebaasi
[[outputs.influxdb]]
  urls = ["http://influxdb:8086"]
  database = "telegraf_deltaEnergy"
  username = "admin"
  password = "admin"
```

Joonis 7. Lisa 1 kasutusmalli alusel loodud konveier Telegrafi platvormil

3.3.2 Automatiseerimine tasemel 1.

Selle peatüki alguses toodud tasemetest esimesel tasemel automatsiooni saavutamise eesmärk ei olnud terviklik ja lõplik lahendus, vaid prototüüp ja ettevalmistus teisel tasemel automatiseerimise projektiks. Sellest tulenevalt tehti otsus kasutada selleks Bash skripti, kuna töö autor oli juba eelnevalt kasutanud erinevaid käsurea utiliite (peamiselt curl[11] ja jq[12]/yq[13]) automatiseerimise alamsammude algsetel katsetustel. Nende kokkupanemisel ühte skripti oli võimalik ehitada, aega kokku hoides, kiire prototüüp, mille põhjal liikuda edasi teise taseme projektini.

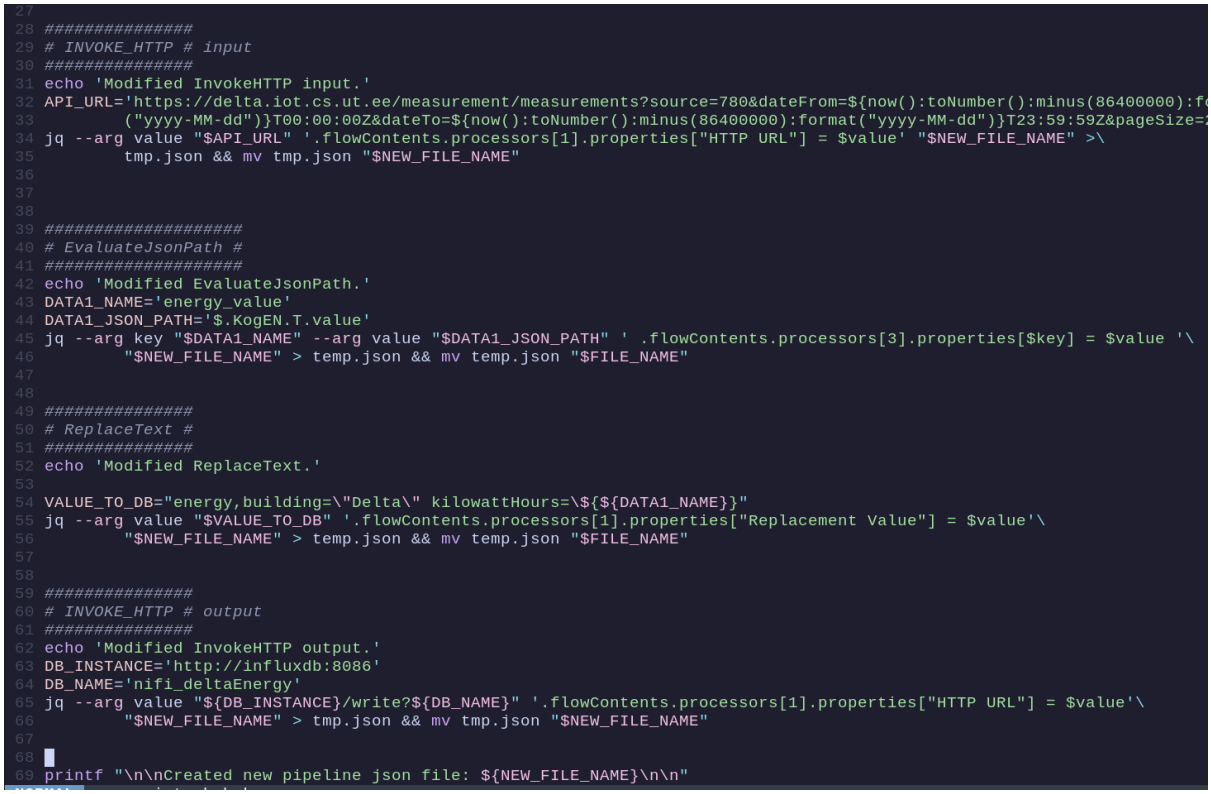
Esimese sammuna oli vaja saavutada loodud andmekonveierite masinloetav kuju, et neid oleks võimalik töödelda. Nifi platvorm võimaldab eksportida enda kasutajaliideses ehitatud konveierid masinloetavasse JSON formaati ning kuna Telegrafi andmekonveiereid defineeritakse TOML formaadis konfiguratsiooni faili, siis ongi see juba masinloetaval kuju.

Kui konveierite masinloetav kuju oli saavutatud, oli järgmine samm kasutada saadud konveiereid mallidena ning kirjutada skript, mis modifitseerib antud malle vastavalt sisendile, ehitades seeläbi uue, soovitud konveieri. Kuna skripti puhul polnud soovi algsest prtotüübist kaugemale minna, siis jäeti parameetrid defineeritavaks skripti sees olevate muutujate kaudu ja skripti jooksmisele kasutajainteraktsiooni juurde ei lisatud. Lisaks praktilisele teadmusele sai pärast selle skripti (vt joonis 8) loomist järeldada, et töö eesmärgiks oleva tasemel 2. automatiseerimise jaoks on vaja luua laiema struktuuriga projekt ning seda on mugavam teha kasutades Pythonit, mitte Bashi.

[11] https://curl.se/

[12] https://jqlang.org/

[13] https://mikefarah.gitbook.io/yq



```
###############
# INVOKE_HTTP # input
###############
echo 'Modified InvokeHTTP input.'
API_URL='https://delta.iot.cs.ut.ee/measurement/measurements?source=780&dateFrom=${now():toNumber():minus(86400000):f
        ("yyyy-MM-dd")}T00:00:00Z&dateTo=${now():toNumber():minus(86400000):format("yyyy-MM-dd")}T23:59:59Z&pageSize=
jq --arg value "$API_URL" '.flowContents.processors[1].properties["HTTP URL"] = $value' "$NEW_FILE_NAME" >\
        tmp.json && mv tmp.json "$NEW_FILE_NAME"



####################
# EvaluateJsonPath #
####################
echo 'Modified EvaluateJsonPath.'
DATA1_NAME='energy_value'
DATA1_JSON_PATH='$.KogEN.T.value'
jq --arg key "$DATA1_NAME" --arg value "$DATA1_JSON_PATH" ' .flowContents.processors[3].properties[$key] = $value '\
        "$NEW_FILE_NAME" > temp.json && mv temp.json "$FILE_NAME"


###############
# ReplaceText #
###############
echo 'Modified ReplaceText.'

VALUE_TO_DB="energy,building=\"Delta\" kilowattHours=\${${DATA1_NAME}}"
jq --arg value "$VALUE_TO_DB" '.flowContents.processors[1].properties["Replacement Value"] = $value'\
        "$NEW_FILE_NAME" > temp.json && mv temp.json "$FILE_NAME"


###############
# INVOKE_HTTP # output
###############
echo 'Modified InvokeHTTP output.'
DB_INSTANCE='http://influxdb:8086'
DB_NAME='nifi_deltaEnergy'
jq --arg value "${DB_INSTANCE}/write?${DB_NAME}" '.flowContents.processors[1].properties["HTTP URL"] = $value'\
        "$NEW_FILE_NAME" > tmp.json && mv tmp.json "$NEW_FILE_NAME"


printf "\n\nCreated new pipeline json file: ${NEW_FILE_NAME}\n\n"
```

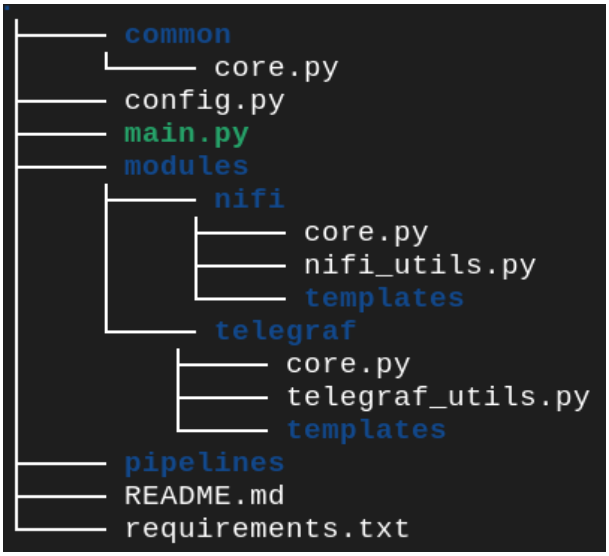
Joonis 8. Taseme 1. saavutamiseks loodud Bashi skripti loogika osa

3.4 Andmekonveieri genereerija loomine

Teisel tasemel automatiseerimise projekti eesmärk oli jõuda selleni, et kuigi väljundallikas on alati sama, antud töö raames influxDB, siis sisendallika komponent on abstrahheeritud - seda on võimalik genereerida vastavalt kasutaja sisendile. Selle saavutamiseks loodi programeerimiskeeles Python projekt, millele pandi projekti alustamisel paika arhitektuur, sooviti saavutada olukord, kus tulevikus on soovi korral lihtne lisada projekti uusi andmekonveieriehitamise platvorme, lisaks selle töö eesmärgiks olnud Nifi ja Telegrafi platvormidele.

3.4.1 Lahenduse disain ja arhitektuur

Arhitektuur, mis algselt paika pandi (vt joonis 9) oli järgmine: projekti juurikas loodi kaust `modules` kus on iga platvormi teegid, mis sisaldavad platvormipõhist funktsionaalust. Korduvkasutatava funktsionaalsuse jaoks loodi teek `common`. Igal moodulil on oma kaust `templates`, kus hoiustatakse andmkonveierite malle - masinloetaval kujul faile, mille hulgast valitakse rakenduse jooksutamisel välja kontekstis soblik fail, mida modifitseeritakse vastavat kasutaja sisendile ja/või konfiguratsioonifailis ette antud väärtuse põhjal, et luua uus andmekonveier.



Joonis 9. Projekti arhitektuur

Juurkausta tekitati ka kaks python faili: `main.py` ja `config.py`, kus esimest kasutatakse skripti käivitamiseks ja platvormi valimiseks ning teist vajalike konfiguratsiooni parameetrite seadmistamiseks. Kuna üks mittefunktsionaalsetest nõuetest oli, et lahendus peab võimaldama täielikult konfiguratsioonifaili põhist seadistamist, siis peab seal olema võimalik defineerida kogu info, mis on vajalik andmekonveieri loomiseks. Nii igaljuhul kohustuslikud parameetrid nagu näiteks andmebaasi autentimisväärtused, aga ka näiteks muutujad, mida saab soovikorral määrata interraktiivselt - näiteks API poolt tagastavate andmete seast soovitud andmeüksuste JSON-i teed.

Lisaks loodi juurkasuta ka failid `README.md` projekti tutvustamiseks ja kasutusjuhendi hoiustamiseks ning `requirements.txt` loenginga Pythoni teekidest, mis on projekti jooksutamiseks vajalikud. Samuti ka kaust `pipelines`, kus hakatakse hoiustatama genereeritud andmekonveiereid.

Tehnoloogiate/teekide valikud

Projekti Python teekide haldamiseks kasutati populaarset pakihaldustööriista pip[14] (*pip Install Packages* akronüüm). Kõik projekti jooksutamiseks vajalikud teegid salvestati *requirements.txt*

[14] https://pypi.org/project/pip/

faili, et need oleks kasutajale teada ja võimalik ühe käsu abil alla tõmmata:
pip install -r requirements.txt.
Järgnevalt tuuakse välja teegid mida kasutati, ning milleks neid kasutati. API kutsete tegemiseks kasutati HTTP päringute jaoks mõeldud de facto teeki `requests`. JSON-andmete parsimiseks kasutati standardteeki `json` ja TOML-andmete parsimiseks kasutati teeki `toml.sys` teeki kasutati, et vajadusel programm kontrollitult lõpetada ning samuti selleks et võimaldada kasutajal anda tööriista jooksutamisel kaasa käsurea parameetreid. Uute failide, millest saavad genereeritavad andmekonveierid, loomiseks kasutati standardteeki `shutils`. Mallide modifitseerimisel kasutati ka teeki `re`, mis võimaldab utiliseerida regexit[15].
Lisaks kasutati ka kahte teeki, `pyfiglet` ja `rich.console`, mis ei andnud projekti funktsionaalsusele midagi juurde, aga mida kasutati tööriista väljundi välimuse muutmiseks.

Mallide loomine

Mallide tekitamiseks loodi kõigepealt vastavatel andmekonveierite ehitamisplatvormidel käsitsi soovitud tüüpi konveier, konverteeriti see siis masinloetavale kujule (kui juba polnud) ning eemaldati osa, mis tegi konveieri eelnevalt eraldiseisvaks ja eristatavaks, nagu näiteks erinevad ID-d, ajatemplid vms. Selliselt tekkisid mallid, millest sai luua erinevaid andmekonveiereid, täites seal vajalikud lüngad nagu näiteks: API url, mille pihta teha andmete päring, päringust filtreeritavad andmed, andmekonveieri nimi jne. Joonisel 10 on toodud näide Telegrafi platvormi mallist.

[15] https://docs.python.org/3/library/re.html#regular-expression-syntax

```
[agent]
  interval = "10s"
  debug = true

[[inputs.http]]
  urls =  []
  method = "GET"
  timeout = "5s"
  headers = { Content-Type = "application/json" }
  data_format = "json"
  json_query = "plcaeholder"
  fieldinclude = []

  name_override = "Placeholder"


[[outputs.influxdb]]
  urls = []
  database =  "Placeholder"
  username = "TODO"
  password = "TODO"
```

Joonis 10. Näide Telegrafi platvormi minimalistlikust ETL konveieri mallist

Nagu juba eelnevalt mainitud siis Nifi ja Telegraf kasutasid konfiguratsioonifailideks ehk andmekonveierite masinloetaval kujul failides hoidmiseks erinevaid faili formaate. Nifi JSON-i formaati ja Telegraf TOML-i formaati. Kusjuures nende failide suurused erinesid kümneid kordi. Lisa 1 all toodud kasutusmalli alusel loodud Telegrafi konfiguratsioonifail oli 35 rida pikk, samas kui Nifi sama kasutusmalli alusel loodud andmekonveieri JSON kujule viidud fail oli 1271 rida pikk.

3.4.2 Tööriista kasutamine

Lahendusena valminud tööriista kasutamiseks on kaks viisi, interaktiivne ja mitteinteraktiivne. Peaükis 3.4.1 „Lahenduse disain ja arhitektuur„ välja toodud failis `config.py` on Boole'i muutuja `INTERACTIVE_MODE`, mida vastavalt kas tõeseks või vääraks muutes saab kasutaja valida, kuidas ta tööriista kasutada soovib. Valides interraktiivse kasutuse, küsitakse kasutajalt kõigepealt mis platvormi ta kasutada soovib ja seejärel API urli (vajadusel ka autentimiseks vajaliku infoga), kust andmeid pärida. Pärast seda kuvatakse päringu vastus kasutajale ning küsitakse milliseid andmeid antud päringust soovitakse filtreerida. Kasutaja saab parsida läbi

päringu (vt joonis 11) ning liikuda samm haaval JSON objektis, kuni soovitud väärtuseni. Seejärel saab kasutaja kas edasi liikuda või valida järgmise väärtuse.

```
Vali json võti või indeks millest soovid väärtuse andmekonveieriga ekstrakteerida

  [0] latitude (float) →  EvaluateJsonPath
  [1] longitude (float) →  EvaluateJsonPath
  [2] generationtime_ms (float) →  EvaluateJsonPath
  [3] utc_offset_seconds (int) →  EvaluateJsonPath
  [4] timezone (str) →  EvaluateJsonPath
  [5] timezone_abbreviation (str) →  EvaluateJsonPath
  [6] elevation (float) →  EvaluateJsonPath
  [7] current_weather_units (dict) →  EvaluateJsonPath
  [8] current_weather (dict) →  EvaluateJsonPath
Vali number (0 - 8): 8
{
  "time": "2025-04-28T17:45",
  "interval": 900,
  "temperature": 9.2,
  "windspeed": 23.4,
  "winddirection": 238,
  "is_day": 1,
  "weathercode": 51
}

Vali json võti või indeks millest soovid väärtuse andmekonveieriga ekstrakteerida

  [0] time (str) →  EvaluateJsonPath
  [1] interval (int) →  EvaluateJsonPath
  [2] temperature (float) →  EvaluateJsonPath
  [3] windspeed (float) →  EvaluateJsonPath
  [4] winddirection (int) →  EvaluateJsonPath
  [5] is_day (int) →  EvaluateJsonPath
  [6] weathercode (int) →  EvaluateJsonPath
Vali number (0 - 6): 2

Valitud väärtus: '.current_weather.temperature'
Oled hetkel valinud järgmised väärtused JSON lõppväärtused:  temperature

Kas soovid (v)alida veel mõne väärtuse või liikuda (e)dasi?(v/e):
```

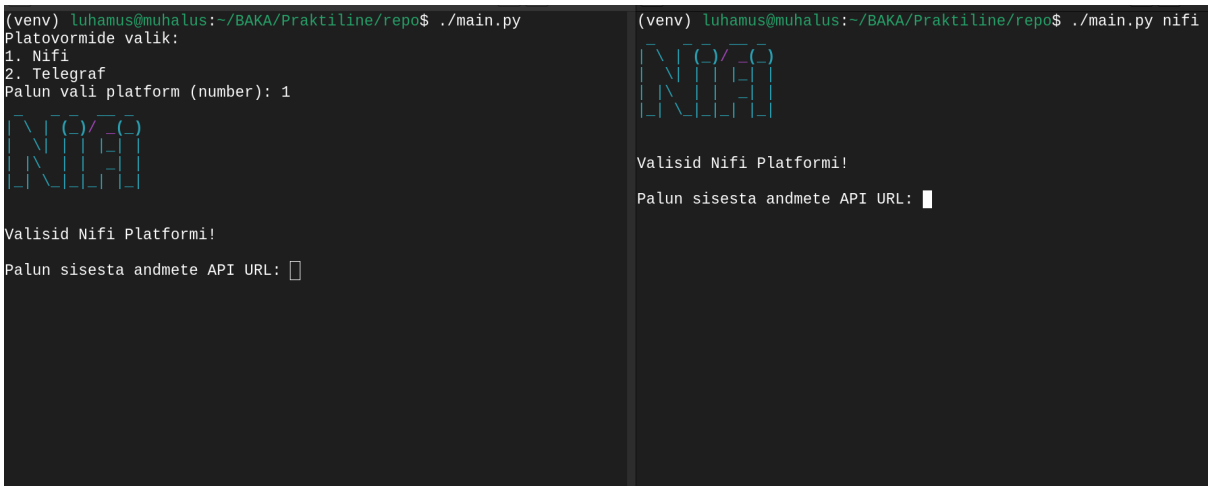
Joonis 11. Lõik tööriista interaktiivses moodis kasutamisest, kus parsitakse tagastatud JSON-it

Pärast seda, kui tööriist on saanud kasutajalt sisendina API urli, ning samuti ka selle tagastavate andmete filtreerimise kohta, siis küsitakse veel ka andmekonveieri nimi ja selle jooksmise sagedus. Samuti küsitakse ka andmebaasi laadimiseks *measurement* nime, mida kasutatakse andmebaasis andmete grupeerimiseks. InfluxDB *measurement* on selgitatud peatükis 2.3.3 „InfluxDB“.

Kui kogu eelnevalt kirjeldatud sisend on kasutajalt vastu võetud, genereeritaksegi malli modifitseerides uus andmekonveier. Kui kasutaja valis Nifi platvormi, siis on ka võimalus

andmekonveier Nifi platvormile paigaldada. Andmekonveieri Nifi platofrmile paigaldamiseks kasutakse Nifi natiivset API-t[16].

Kui kasutaja valib aga mitteinteraktiivse viisi tööriista kasutamiseks, siis peab kogu eelnev info olema enne tööriista jooksutamist defineeritud konfiguratsioonifailis `config.py`, kust see siis võetakse ja mille abil andmekonveier genereeritakse. Tööriist võimaldab konfiguratsioonifailis defineerida ka platvormi mida kasutatakse, tagades sellega, et mitteinteraktiivses moodis on tööriista võimalik kasutada ka näiteks skriptides, kuna ühtegi sisendit pole pärast käivitamist enam vaja võtta. Samuti on võimalik valida platvorm läbi käsureamuutuja (vt joonis 12).



Joonis 12. Töörista käivitamisel saab platvormi valida ka läbi käsureamuutuja

Lisaks, hoolimata sellest kas on valitud interaktiivne või mitteinteraktiivne kasutusviis, on muutujad, mis tuleb enne tööriista kasutamist konfiguratsioonifailis defineerida. Selleks on andmebaasi URL, kasutajanimi ja parool. Kogu info tööriista kasutamise ja konfiguratsioonifaili väärtuste kohta on kasutaja jaoks toodud välja failis *README.md*, mis on soovituslik enne tööriista kasutamist läbi lugeda.

Järgmine peatükk katab valminud tööriista kasutamise ja selle testimise ning sellest saadud tagasiside. Samuti probleemid ja võimalused töö edasiarendamiseks.

[16] https://nifi.apache.org/nifi-docs/rest-api.html

4. Testimine ja analüüs

Selles peatükis tuuakse välja valminud lahenduse valideerimise ja testimise protsess ning selle tulemused.

4.1 Lahenduse nõuete kontrollimine

Lahenduse testimise aluseks võeti peatükis 3.2 „Lahenduse nõuded“ all välja toodud funktsionaalsed- ja mittefunktsionaalsed nõuded (edaspidi FN ja MFN).

Esimene kasutusmall (vt Lisa 2) mida testmiseks kasutati, kujutas olukorda, kus on loend riike (Eesti, Läti, Leedu, Poola, Saksamaa, Holland, Hispaania, Portugal, Itallia, Kreeka). Igale riigile soovitakse teha oma andmekonveier, mis küsib selle riigi pealinna kohta ilma andmeid, filtreerib siis API vastusest soovitud andmed ja salvestab need etteantud asukohta. API mida selle jaoks kasutati oli Open-Meteo API. Andmed mida vastusest soovitakse kätte saada on temperatuur ja tuule kiirus. Lõpp-punkt on tööriista poolt fikseeritud InfluxDB.

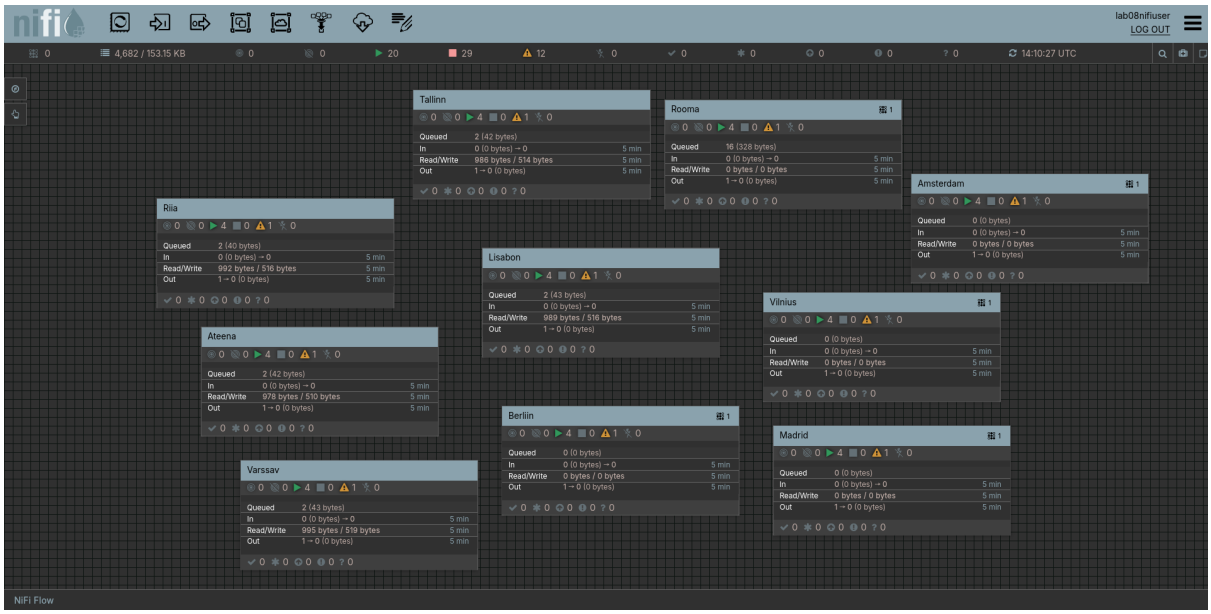
Kasutusmalli läbimiseks loodi Python skript (vt joonis 13), kus antakse ette loend linnadest mille kohta andmekonveiereid soovitakse ning siis kasutatakse selle töö raames valminud tööriista, et need genereerida. Skript muudab iga linna jaoks konfiguratsioonifailis API otspunkti, andmekonveieri nime ning andmebaasi minevate andmete eristamiseks mõeldud lõpp-punkti (ingl *measurement*) nime.

Ülejäänud sisend, nagu platvormi valik, andmekonveierite jooksmise tihedus, soov andmekonveierid ka platvormile paigaldada, platvormi host ja autentimise andmed, andmebaasi andmed ning andmed mida soovitakse filtreerida on defineeritud konfiguratsioonifailis ja neid ei pea iga linna kohta eraldi muutma. Lisaks lülitati tööriist mitteinteraktiivsesse moodi, mis on skripti jooksutamiseks oluline.

```python
import re
import subprocess

locations = [
    {"riik": "Eesti", "pealinn": "Tallinn", "lat": 59.4370, "lon": 24.7536},
    {"riik": "Läti", "pealinn": "Riia", "lat": 56.9496, "lon": 24.1052},
    {"riik": "Leedu", "pealinn": "Vilnius", "lat": 54.6872, "lon": 25.2797},
    {"riik": "Poola", "pealinn": "Varssav", "lat": 52.2297, "lon": 21.0122},
    {"riik": "Saksamaa", "pealinn": "Berliin", "lat": 52.5200, "lon": 13.4050},
    {"riik": "Holland", "pealinn": "Amsterdam", "lat": 52.3676, "lon": 4.9041},
    {"riik": "Hispaania", "pealinn": "Madrid", "lat": 40.4168, "lon": -3.7038},
    {"riik": "Portugal", "pealinn": "Lisabon", "lat": 38.7169, "lon": -9.1399},
    {"riik": "Itaalia", "pealinn": "Rooma", "lat": 41.9028, "lon": 12.4964},
    {"riik": "Kreeka", "pealinn": "Ateena", "lat": 37.9838, "lon": 23.7275}
]

with open("config.py", "r", encoding="utf-8") as file:
    original_config = file.read()

for el in locations:
    new_url = f"https://api.open-meteo.com/v1/forecast?latitude={el['lat']}&longitude={el['lon']}&current_weather=true"

    updated_config = re.sub( r'API_URL\s*=.*', f'API_URL="{new_url}"', original_config)
    updated_config2 = re.sub( r'MEASUREMENT_NAME\s*=.*', f'MEASUREMENT_NAME="{el["pealinn"]}_ilm"', updated_config)
    updated_config3 = re.sub( r'PIPELINE_NAME\s*=.*', f'PIPELINE_NAME="{el["pealinn"]}"', updated_config2)


    with open("config.py", "w", encoding="utf-8") as file:
        file.write(updated_config3)

    print(f"Genereerin andmekonveierit {el["riik"]} pealinnale.")
    subprocess.run(["python", "main.py"], check=True)
```

Joonis 13. Andmekonveierite genereemiseks kirjutatud skript

Pärast selle skripti edukat jooksutamist tekkiski Nifi platvormile 10 uut andmekonveierit (vt joonis 14), mis olid esmalt genereeritud JSON kujule ning seejärel laetud Nifi API abil platvormile ka ülesse. Pärast konveierite genereerimist, paigaldamist ja käima panemist oli võimalik veenduda, et konveierid ka töötavad kontollides, kas andmed laekuvad ka andmebaasi (vt joonis 15).



Joonis 14. Pärast joonisel 13 toodud skripti käivitamist tekkisid Nifi platvormile 10 uut andmekonveierit

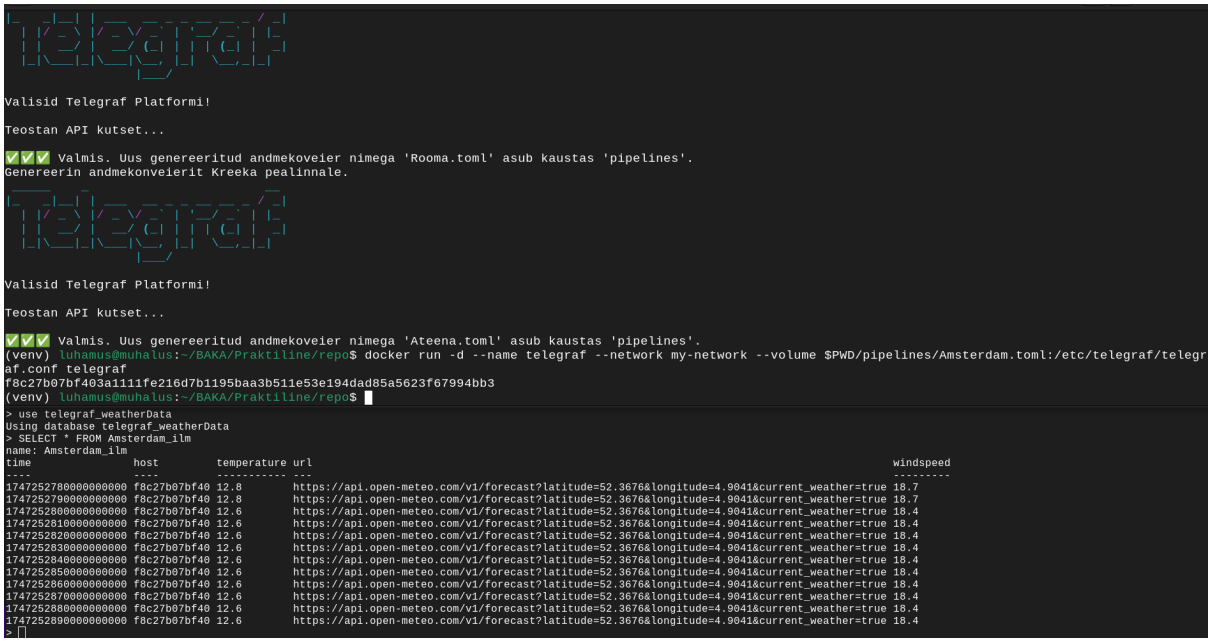
```
luhamus@muhalus:~$ docker exec -it influxdb influx -database 'nifi_weatherData'
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> show measurements
name: measurements
name
----
Amsterdam_ilm
Ateena_ilm
Berliin_ilm
Lisabon_ilm
Madrid_ilm
Riia_ilm
Rooma_ilm
Tallinn_ilm
Varssav_ilm
Vilnius_ilm
> select * from Rooma_ilm
name: Rooma_ilm
time                temperature windspeed
----                ----------- ---------
1746367008162861183 22          18.2
1746367018195488519 22          18.2
1746367028249461466 22          18.2
1746367038301785810 22          18.2
1746367048364109347 22          18.2
```

Joonis 15. Pärasti joonisel 14 toodud andmekonveierite käivitamist tekkisid andmebaasi iga konveieri kohta ka andmed

Seega saab öelda, et kasutusmall viidi edukalt läbi. Lisaks saab öelda, et kirja pandud nõuetest täideti selles kasutusmallis järgmised: FN1, FN3, FN4, FN6, FN7 (v.a neljas alampunkt), FN8, FN9. Kuna FN10, FN11 ja FN12 pole selle kasutusmalliga valideeritavad siis jäävad need praegu kõrvale, valideerida on veel vaja FN2 ja FN5.

FN5, ehk interaktiivse kasutusmoe toetuse kontrollimiseks jooksutati andmekonveierit interaktiivses moodis ning valiti api otspunktiks kümnest pealinnast suvaliselt üks, Ateena. Pärast interaktiivse kasutusmoega sisendite andmist ja andmekonveieri Nifi platvormile paigaldamist sai veenduda, et ka interaktiivne mood töötab ehk FN5 võis lugeda täidetuks.

Selleks et kontrollida FN2 (tööriist võimaldab valida toetatud platvormide vahel) muudeti konfiguratsioonifailis platvormi parameeter Nifilt Telegraafile. Pärast eelnevalt välja toodud skripti (vt joonsi 13) uuesti jooksutamist genereeriti nüüd samamoodi 10 andmekonveierit, aga mitte Nifi vaid Telegrafi platvormile ja seega mite JSON kujul vaid TOML kujul. Pärast andmekonveierite jooksutamist Telegrafiga tekkisid samamoodi andmekonveieri poolt saadetud andmed influxDB andmebaasi. Seda on demostreeritud joonisel 16.



Joonis 16. Joonisel 13 toodud skripti abil andmekonveierite genereerimine Telegrafile ning nende korrektsuse kontrollimine

MF2, ehk omadus, et lahendus on deterministlik ning mugavalt kasutatav skriptimises, on samuti selle eelnevate katsetega kaetud. Lisaks ka MFN3, ehk nõue, et kui programmi jooksmise käigus tekib probleem peab kasutajale tagastama veateate (vt joonis 17).

```
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ python main.py
Platovormide valik:
1. Nifi
2. Telegraf
Palun vali platform (number): vale
Ebaõnnestunud valik, sulgen rakenduse...
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ python main.py nifi
 _   _ _  __ _
| \ | (_)/ _(_)
|  \| | | |_| |
| |\  | |  _| |
|_| \_|_|_| |_|


Valisid Nifi Platformi!

Palun sisesta andmete API URL: https://www.ebakorrektneURL.xyz
Kas API vajab ka kasutajaga autentimist?(jah/ei): ei
Teostan API kutset...

HTTP error: HTTPSConnectionPool(host='www.ebakorrektneurl.xyz', port=443): Max retries exceeded with url: / (Caused by NameResolutionError("<urllib3.connection.HTTPSConnection object at 0x7f02838ff950>: Failed to resolve 'www.ebakorrektneurl.xyz' ([Errno -2] Name or service not known"))

Kas soovid URL-i (m)uuta URL-i või (v)äljuda?(m/v):
```

Joonis 17. Näited MFN3 täitmisest

FN10 on täidetud läbi projektis oleva `README.md` faili (vt Lisa 5), kus on kasutajale vajalik dokumentatsioon, kus tutvustatakse tööriista ja antakse juhised kasutamiseks vajaliku keskkonna ettevalmistamiseks ning konfiguratsioonifaili modifitseerimiseks

FN11, ehk projekti modulaarne strukturiseeritus, on kaetud tänu arhitektuuri valikule projekti

alguses. Iga platvorm on eraldi teek ja ühiskasutatav funktsionaalsus on teegis `common`. Peatükis 3.4.1 „Lahenduse disain ja arhitektuur“, on ka joonis (vt joonis 9) projekti modulaarsest arhitektuurist.

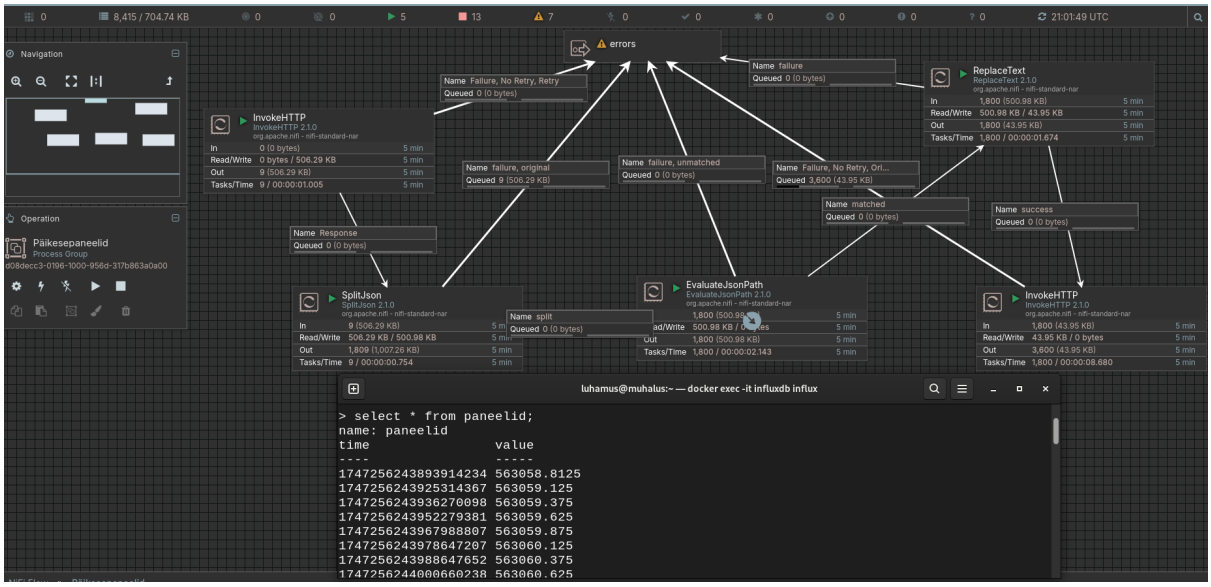
Seega testimist vajas veel MFN1, ehk erinevate sisendallikate API-de toetus. Samuti kasutajanime ja parooli sisendi toetatavus, ehk FN12 ning FN7 neljas alampunkt. Viimaste kontrollimiseks oli vaja kasutada eelnevast kasutusmallist erinevat API-t, mis omakord kontrollib ka MFN1.

Nende nõuete kontrollimiseks võetigi kasutusele teine arendusel kasutatud API, Delta päikesepaneelide andmete API. Tööriista jooksutati interaktiivses moodis eesmärgiga genereerida andmekonveier, mis saab API-st mõne konkreetse päeva andmed, filtreerib sealt info päikesepaneelide kilovatt-tundide tootlikuse kohta ja laeb selle andmebaasi ülesse. Enne tööriista jooksutamist sisestati konfiguratsioonifaili andmebaasi info ning muudeti tööriista mood interaktiivse peale. Pärast sisendi andmist, andmekonveieri genereerimist, paigaldamist ja käivitamist tekkisid andmed ka andmebaasi ning seega oli vastavate nõuete kontroll edukas. Joonistel 18 ja 19 on näha konveieri genereerimist ja kontrollimist.

```
  [5] KogEN (dict) →  EvaluateJsonPath
Vali number (0 - 5): 5
{
  "T": {
    "unit": "kilowattHours",
    "value": 563058.8125
  }
}

Vali json võti või indeks millest soovid väärtuse andmekonveieriga ekstrakteerida

  [0] T (dict) →  EvaluateJsonPath
Vali number (0 - 0): 0
{
  "unit": "kilowattHours",
  "value": 563058.8125
}

Vali json võti või indeks millest soovid väärtuse andmekonveieriga ekstrakteerida

  [0] unit (str) →  EvaluateJsonPath
  [1] value (float) →  EvaluateJsonPath
Vali number (0 - 1): 1

Valitud väärtus: '.measurements[0].KogEN.T.value'
Oled hetkel valinud järgmised väärtused JSON lõppväärtused:  value

Kas soovid (v)alida veel mõne väärtuse või liikuda (e)dasi?(v/e): e
{'value': '.measurements[0].KogEN.T.value'}

Kui tihti peaks andmekonveier jooksma? (sekundites)
Vali number (0 - 86400): 30
Mis saab andmekonveieri nimeks: Päikesepaneelid
Palun sisesta andmebaasi jaoks vajalik 'measurement' nimi (influxDB): paneelid
✅✅✅ Valmis. Uus genereeritud andmekoveier nimega 'Päikesepaneelid.json' asub kaustas 'pipelines'.
✅ Andmekonveier 'Päikesepaneelid' on edukalt ka Nifi platvormile paigaldatud!
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$
```

Joonis 18. Päikesepaneelide API konveieri interkatiivses moodis genereerimine

MFN1 lõplikuks kontrollimiseks prooviti tööriista kasutada veel kahe uue API-ga, mida arenduse käigus ei kasutatud. Nende leidmiseks otsiti interntist tasuta API-sid ning valiti välja kaks:



Joonis 19. Genereeritud Päikesepaneelide API konveieri kontrollimine

ExchangeRate API [17] ja OpenWeather API [18]. Esimese, ExchangeRate API puhul kasutati genereerimiseks interaktiivset moodi ja teise, OpenWeather API puhul mitteinteraktiivset moodi, ehk siis anti sisend konfiguratsioonifaili kaudu. Sellega taaskontrolliti ka FN5, ehk võimekust kasutada tööriista nii interaktiivselt kui ka mitteinteraktiivselt.
Lisaks, selleks et taaskontrollida ka FN2, ehk võimekust toetada mitut platvormi, genereeriti ExchangeRate API andmekonveier Nifi platvormile, OpenWeather API konveier aga Telegrafi platvormile.

ExchangeRate API puhul kasutati interaktiivset kasutusmoodi ning eesmärgiks oli genereerida andmekonveier, mis tõmbab Dollari ja Euro vahelise valuutakursi. Andmekonveieri genereerimise ja korrektsuse kontrollimise sammudest on toodud joonised Lisa 3 all.

OpenWeather API puhul kasutati andmekonveieri loomiseks juba olemasolevat kasutusmalli (vt Lisa 2) Tartu linna kohta.
OpenWeather API konveieri genereerimiseks väärtustati esmalt konfiguratsioonifailis järgmised väärtused:

[17] https://www.exchangerate-api.com/
[18] https://openweathermap.org/

```
INTERACTIVE_MODE=False
PLATFORM="Telegraf"
MEASUREMENT_NAME="Tartu_ilmaandmed"
DB_URL="http://influxdb:8086"
DB_NAME="telegraf_weatherData"
DB_USER="admin"
DB_PASS="admin"
API_URL="https://api.openweathermap.org/data/2.5/weather
         ?q=Tartu&units=metric&lang=en&appid=01786b7e8b6
         23a1d2112d672ecae1d0d"
API_FIELDS={'temp': '.main.temp', 'winds': '.wind.speed'}
PIPELINE_SCHEDULING_PERIOD="10"
PIPELINE_NAME="OpenWeather_pipeline"
```

Seejärel genereeriti konveier, ning pandi see kontrollimiseks käima. Andmebaasi tekkisid uued andmed (vt joonis 20) ning seega loeti uue API-ga konveieri genereerimise katse edukaks.

```
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ python main.py
 _____    _                        __
|_   _|__| | ___  __ _ _ __ __ _ / _|
  | |/ _ \ |/ _ \/ _` | '__/ _` | |_
  | |  __/ |  __/ (_| | | | (_| |  _|
  |_|\___|_|\___|\__, |_|  \__,_|_|
                 |___/

Valisid Telegraf Platformi!

✅✅✅ Valmis. Uus genereeritud andmekoveier nimega 'OpenWeather_pipeline.toml' asub kaustas 'pipelines'.
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ docker run -d --name telegraf --network my-network --volume $PWD/
pipelines/OpenWeather_pipeline.toml:/etc/telegraf/telegraf.conf telegraf
99d8050e9285a874ac230cfabe22fd18d596705ccc93a7162b3e3bca0aea79d3
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ docker exec -it influxdb influx --database 'telegraf_weatherData'

Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> select * from Tartu_ilmaandmed;
name: Tartu_ilmaandmed
time                host         main_temp url
                                           wind_speed
----                ----         --------- ---
                                           ----------
1746966931000000000 99d8050e9285 9         https://api.openweathermap.org/data/2.5/weather?q=Tartu&units=metric&
lang=en&appid=01786b7e8b623a1d2112d672ecae1d0d 5.14
1746966940000000000 99d8050e9285 9         https://api.openweathermap.org/data/2.5/weather?q=Tartu&units=metric&
lang=en&appid=01786b7e8b623a1d2112d672ecae1d0d 5.14
1746966950000000000 99d8050e9285 9         https://api.openweathermap.org/data/2.5/weather?q=Tartu&units=metric&
lang=en&appid=01786b7e8b623a1d2112d672ecae1d0d 5.14
```

Joonis 20. OpenWeather API andmekonveieri geneerimine ja jooksutamine

4.2 Nifi ja Telegrafi võrdlus

Töös välja valitud platvormide vahel oli mitmeid erinevusi. Alates väljanägemisest ja kasutuskeskkonnast kuni funktsionaalsuse ja andmekonveierite formaadini. Järgnevalt võrreldakse Nifit ja Telegrafi ning tuuakse välja mitmeid erinevusi, mida autor töö jooksul

täheldas, aga mis on seega kohati ka subjektiivsed.

Kasutajaliides

Nagu juba eelevalt mainitud, siis on Nifil ja Telegrafi vundamentaalselt erinevad kasutajaliidesed: Nifil on graafiline ja brauseripõhine kasutajaliides, Telegrafil aga käsureapõhine. Nifi on kahest platvormist kindlasti võimaluste rikkam, ning seega ka komplektsem ja subjektiivselt keerulisem. Autoril läks Nifi platvormiga tutvumiseks umbes kaks korda rohkem aega kui Telegrafiga. Nifi jaoks on arendajate poolt välja töötatud ka selle natiivne API, mis võimaldab platvormiga suhelda. Selle töö raames kasutatakse seda API-t pärast andmekonveierite loomist nende platvormile paigaldamiseks.

Andmekonveierite ehitamine

Nifi andmekonveierite ehitamine on mõeldud läbi graafilise liidese, kus ükshaaval konfigureeritakse erinevaid protsessoreid, kuni neist saab kokku üks andmekonveier. Telegrafi puhul tähendab andmekonveieri ehitamist konfiguratsiooni faili kirjutamine. Viimases defineeritakse soovitud pistikprogrammid (ing *pluggin*), mis on ekivalentsed Nifi protsessoritega ja mis samamoodi kokku teevad andmekonveieri. Telegrafi konfiguratsiooni failide defineerimiseks kasutatakse TOML keelt. Nifi peab graafilises liideses defineeritud andmekonveiereid kuidagi ka salvestama ja seda tehakse JSON failidesse, mida kasutajal on võimalik ka kätte saada (eksportida). Seega on võimalik ka Nifi konveiereid ehitada läbi konfiguratsiooni defineerimise, kuigi see pole eelistatud viis.

Lisaks, nagu juba peatüki 3.4.1 alampunktis “Mallide loomine” välja toodud, siis kui võrrelda sama kasutusmalli alusel defineeritud konfiguratsioonifaili kujul andmekonveiereid, siis on nende suurusel märgatav vahe.

Planeeritud kasutusjuhud

Platvormide arendusel on neile planeeritud ka veidi erinevad kasutusjuhud. Mõlema peamine eesmärk on küll andmekonveierite ehitamine ja jooksutamine, kuid Nifi platvormi puhul on pandud rohkem rõhku ja mugavust ka pärast konveierite valmimist nende haldamisele [23]. Nifi platormil on võimalik luua erinevaid rolle ning määrata neile õiguseid, samuti saab seal salvestada üle platvormi kasutatavaid muutujaid. Nagu eelnevalt mainitud siis on haldamiseks loodud ka eraldi API, samuti võib graafilist liidest võtta kui andmekonveierite haldamist mugavdavat elementi. Võrreldes Nifiga, Telegraf selliseid mugavusi ei võimalda. See tuleb välja näiteks ka andmekonveierite käivitamisel ja peatamisel, kus Nifi puhul on graafilises liidesest mugavad

“Start” ja “Stop” nupud, aga Telegrafi puhul tuleb andmekonveieri käivitamiseks ja peatamiseks vastavalt kas panna käima või peatada kogu Telegrafi agent. Ka monitoorimine ja silumine (ingl *debugging*) on Nifi puhul võimekam. Iga protsessori kohta on eraldi detailsed logid, samuti on logid ka tervikliku protsessorite grupi kohta. Telegrafi puhul on nö lihtsalt agendi logid, kus peal konveier jookseb. Samuti pole sealsed logid nii detailsed.

Viimasena, kui kasutajal tekib soov midagi andmekonveieri konfiguratsioonis muuta, siis Telegrafi puhul tuleb selleks kogu agent seisma panna, teha konfiguratsioonifailis muudatused ning siis uuesti agent käivitada. Nifi puhul on muudatuste tegemine aga dünaamilisem ja muudatusi on võimalik teha protsessoriti ka samal ajal, kui ülejäänud andmekonveier jookseb.

4.3 Töö edasi arendamine

Juba töö praktilise osa alguses pandi eesmärgiks, et valminud lahendust oleks võimalik tulevikus täiendata. Seega üks võimalus töö edasi arendamiseks on kindlasti uue platvormi või platvormide lisamine. Samuti olemasolevatele platvormidele erinevate uute mallide lisamine.

Samuti saaks arendada tööriista selliselt et see tõuseks töö käigus paikapandud tasemetest teiselt tasemelt viimasele, kolmandale. See tähendaks seda, et tööriist võimaldaks valida ka lõpp-punkti, ehk andmebaasi. Praegu on see fikseeritud.

Tehnilise poole pealt võiks kindlasti parandada paroolide ja muu tundliku info haldamist, neid näiteks krüpteerides, kuna praegune lahendus seda ei tee. Telegraf vaikimisi mingisugust tundlikute andmete varjamist ei toeta, aga selle lahendamist võiks proovida näiteks läbi keskkonnamuutujate või ka tuues sisse mõne lisa tööriista, näiteks HashiCorp Vault[19]. Nifi tegelikult toetab teatud määral andmete krüpteerimist [24]. Seda sisseehitatud toetust saaks proovida kasutada et koos Nifi natiivse API-ga kasutaks ka tööriist krüpteeritud paroole. Tööd võiks ka veelgi erinevate API-dega testida.

[19] https://developer.hashicorp.com/vault

5. Kokkuvõte

Selle Bakalaureusetöö eesmärgiks oli uurida erinevaid andmekonveierite ehitamise platvorme ning arendada seejärel välja lahendus mis on võimeline genereerima valitud platvormidele sobivaid andmekonveiereid, aidates seeläbi automatiseerida andmekonveierite ehitamist.

Töö algas tutvustusega sellest, mis on andmekonveierid ning kas ja kuidas on eelnevalt proovitud andmekonveierite ehitamise automatiseerimist lahendada. Seejärel tutvustati välja valitud andmekonveierite platvorme ning samuti praktilise lahenduse jaoks kasutatavaid tehnoloogiaid.

Pärast seda anti ülevaade praktilise lahenduse valmimisest. Kirjeldati esimesi prototüüpe ning katsetusi, samuti lahenduse testimiseks kirja pandud nõudeid. Lahendusena valmis tööriist, mis võtab kasutajalt andmekonveierite konfigureerimise kohta sisendi ning genereerib seejärel kasutajale soovitud andmekonveieri. Nifi platvormi puhul paigaldab tööriist konveieri ka platvormile. Tööriista lõpp-punkt on fikseeritud, influxDB. Andmeallika API on aga võimalik vabalt valida. Toetatud on interaktiivne ja mitteinteraktiivne töömood.

Viimases peatükis testiti valminud lahendust erinevate kasutusmallide abil, võttes aluseks eelnevalt paika pandud nõuded. Testimise järgselt sai väita, et paika pandud nõuded said täidetud. Valminud tööriista võiks lihtsustada ja kiirendada andmekonveierite ehitamise ja paigaldamise protsessi.

Viimases peatükis võrreldi ka töö jaoks välja valitud andmekonveierite platvorme, Nifit ja Telegraafi. Viimaks toodi ka ülevaade töö edasise arendamise võimalikest sammudest.

Viited

[1] Densmore J. Data Pipelines Pocket Reference: Moving and Processing Data for Analytics. 1st edition. O'Reilly Media, 2021.

[2] Paul C. ja Smith M. Optimizing Data Pipelines with Advanced ETL Automation Techniques. *ResearchGate* (detsember 2022). PDF Available. https://www.researchgate.net/publication/387534348_Optimizing_Data_Pipelines_with_Advanced_ETL_Automation_Techniques.

[3] InfluxData. Telegraf - Open Source Server Agent for Collecting Metrics. https://www.influxdata.com/time-series-platform/telegraf/ (09.03.2025).

[4] Apache Software Foundation. Apache NiFi - An Integrated Data Logistics Platform for Automating Data Movement. https://nifi.apache.org/ (09.03.2025).

[5] Elementl. Dagster - Data Orchestration Tool. https://dagster.io/ (18.04.2025).

[6] Gupta M. Orchestration and Standardization of Data Pipelines using TOSCA. MSc Thesis. Tallinn University of Technology, 2022. https://thesis.cs.ut.ee/f7bca3a3-15d0-445e-b2a2-d3a39329c1c0.

[7] Mbata A., Sripada Y. ja Zhong M. A Survey of Pipeline Tools for Data Engineering. 2024. https://doi.org/10.48550/arXiv.2406.08335.

[8] Donca I.-C., Stan O. P., Misaros M., Gota D. ja Miclea L. Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects. *Sensors* 22.12 (2022). https://doi.org/10.3390/s22124637.

[9] Valdas A. ML-TOSCA: ML pipeline modelling and orchestration using TOSCA. MSc Thesis. University of Tartu, 2023. https://thesis.cs.ut.ee/7de885b2-a6d4-456c-a834-6e8f9db330ce.

[10] Chilukoori V. V. R. Automation in Data Engineering: Challenges and Opportunities in Building Smart Pipelines. *ResearchGate* (2024). https://www.researchgate.net/profile/Vishnu-Vardhan-Reddy-Chilukoori/publication/388647391_Automation_in_Data_Engineering_Challenges_and_Opportunities_in_Building_Smart_Pipelines/links/67a0f5d58311ce680c4e19d2/Automation-in-Data-Engineering-Challenges-and-Opportunities-in-Building-Smart-Pipelines.pdf.

[11] Synergy Research Group. Cloud Market Jumped to 330 billion in 2024 – GenAI is Now Driving Half of the Growth. https://www.srgresearch.com/articles/cloud-market-jumped-to-330-billion-in-2024-genai-is-now-driving-half-of-the-growth (15.05.2025).

[12] Google Cloud. Open Source ETL Pipeline Tool - Google Cloud Data Fusion. https://cloud.google.com/blog/products/data-analytics/open-source-etl-pipeline-tool (16.03.2025).

[13] Microsoft Azure. Build an End-to-End Data Pipeline in Databricks. Microsoft. https://learn.microsoft.com/en-us/azure/databricks/getting-started/data-pipeline-get-started (16.03.2025).

[14] Amazon Web Services. Implementing Dynamic ETL Pipelines Using AWS Step Functions. https://aws.amazon.com/blogs/compute/implementing-dynamic-etl-pipelines-using-aws-step-functions/ (16.03.2025).

[15] Oleghe O. ja Salonitis K. A framework for designing data pipelines for manufacturing systems. *Procedia CIRP* 93 (2020). 53rd CIRP Conference on Manufacturing Systems, lk 724–729. https://doi.org/10.1016/j.procir.2020.04.016.

[16] Apache NiFi. Apache NiFi Processor Documentation. Apache Software Foundation. https://nifi.apache.org/docs.html (18.04.2025).

[17] Apache NiFi. Apache NiFi Expression Language Guide. https://nifi.apache.org/docs/nifi-docs/html/expression-language-guide.html (23.03.2025).

[18] InfluxData. InfluxData Platform. https://www.docs.influxdata.com (28.03.2025).

[19] InfluxData. Telegraf Documentation Introduction. https://docs.influxdata.com/telegraf/v1/get-started/ (20.04.2025).

[20] InfluxData. InfluxDB Documentation. https://docs.influxdata.com/influxdb3/core/ (03.04.2025).

[21] InfluxData. Line Protocol Tutorial. https://docs.influxdata.com/influxdb/v1/write_protocols/line_protocol_tutorial/ (28.04.2025).

[22] Open-Meteo. Open-Meteo: Free Weather API for Non-Commercial Use. https://open-meteo.com/ (18.04.2025).

[23] Apache NiFi. NiFi System Administrator's Guide. https://nifi.apache.org/docs/nifi-docs/html/administration-guide.html (15.05.2025).

[24] Apache NiFi. NiFi System Administrator's Guide. https://nifi.apache.org/docs/nifi-docs/html/administration-guide.html%5C#sensitive_flow_migration (15.05.2025).

Lisad

Lisa 1. Kasutusmall käsitsi andmekonveierite loomiseks Nifi ja Telegrafi platvormidel

”Dünaamilise Andmekonveieri Loomine”

Osalised: admin, andmekonveierite platvormid, Delta maja päikesepaneelide andmete API, Influx andmebaas

Eesmärk: Luua dünaamiline andmekonveier, mis tõmbab API-st alati eilse päeva päikesepaneelide andmed (kilovatt-tundide toodang), filtreerib need ja laeb ülesse andmebaasi.

Sammud:

- Admin seadistab andmekonveieri.
- Andmekonveier käivitatakse.
- Andmekonveier tõmbab API-st andmed.
- Andmekonveier filtreerib andmed vastavalt admini seadistustele.
- Andmekonveier laeb filtreeritud andmed andmebaasi.

Eeldustingimused:

- Andmekonveierite platvormid on paigaldatud ja valmis kasutamiseks.
- Andmebaas on eelnevalt seadistatud ja jooksma pandud.

Lisa 2. Kasutusmall mitme riigi andmekonveierite dünaamiliseks genereerimiseks

”Mitme Riigi Ilmaandmete Automaattöötlus”

Osalised: Admin, andmekonveierite tööriist, Open-Meteo ilmaandmete API, Influx andmebaas.

Eesmärk: Luua iga soovitud riigi pealinna jaoks eraldi andmekonveier, mis hangib temperatuuri ja tuulekiiruse andmed vastavast API-st ning salvestab need määratud andmebaasi.

Sammud:

- Admin määrab sihtriigid, nende pealinnad ja koordinaadid.
- Tööriist genereerib iga riigi jaoks unikaalse API-päringu, kasutades vastavat koordinaati.
- Iga riigi kohta luuakse eraldi andmekonveier, mis käivitatakse automaatselt.
- Andmekonveier hangib vastava riigi ilmastikuandmed API-st.
- Andmekonveier filtreerib välja ainult vajaliku info: temperatuur ja tuulekiirus.
- Filtreeritud andmed salvestatakse andmebaasi riigipõhise identifikaatoriga.

Eeldustingimused:

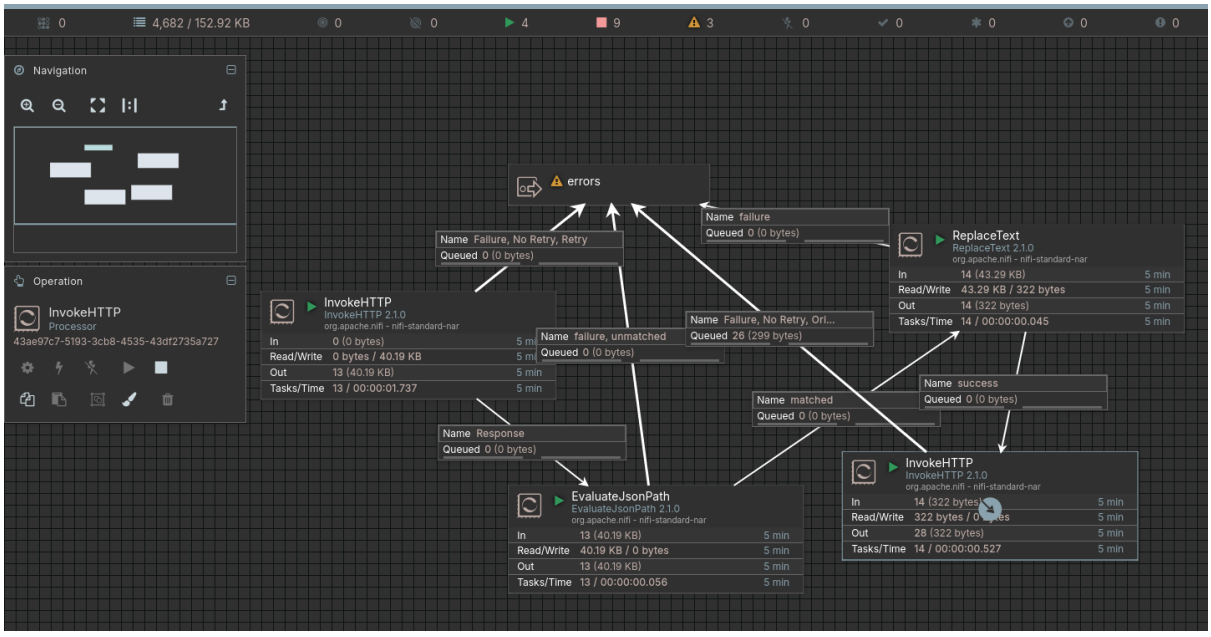
- Eksiteerib andmekonveierite genereerimise tööriist ja see toetab automaatset töövoogu.
- InfluxDB andmebaas on paigaldatud ja ligipääsetav.
- API on ligipääsetav ja võimeline tagastama andmeid iga riigi koordinaatide kohta.

Lisa 3. ExchangeRate API testimise sammud

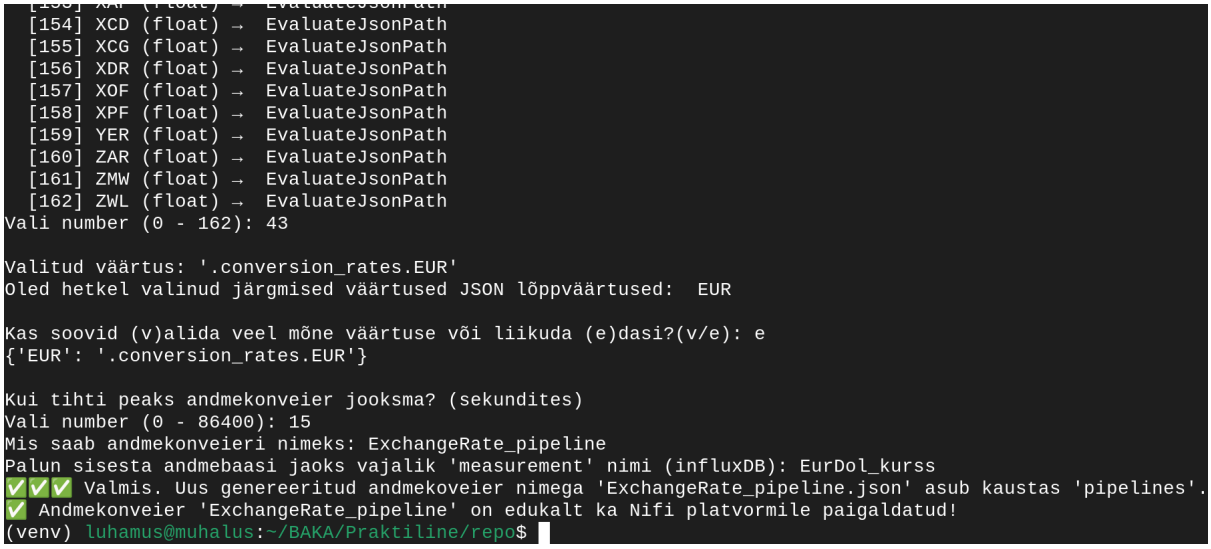
Siin on toodud sammud testimisest ExchangeRate API-ga, millest on kirjutatud peatükis 4.1 “Nõuete kontroll”.

```
(venv) luhamus@muhalus:~/BAKA/Praktiline/repo$ python main.py
Platovormide valik:
1. Nifi
2. Telegraf
Palun vali platform (number): 1
 _    _ _   __ _
| \  | (_)/ _(_)
|  \| | | |_| |
| |\  | |  _| |
|_| \_|_|_| |_|


Valisid Nifi Platformi!

Palun sisesta andmete API URL: https://v6.exchangerate-api.com/v6/78660310eae8ed2c9ab662f8/latest/USD
Kas API vajab ka kasutajaga autentimist?(jah/ei): ei
Teostan API kutset...

{
  "result": "success",
  "documentation": "https://www.exchangerate-api.com/docs",
  "terms_of_use": "https://www.exchangerate-api.com/terms",
  "time_last_update_unix": 1746921601,
  "time_last_update_utc": "Sun, 11 May 2025 00:00:01 +0000",
  "time_next_update_unix": 1747008001,
  "time_next_update_utc": "Mon, 12 May 2025 00:00:01 +0000",
  "base_code": "USD",
  "conversion_rates": {
    "USD": 1,
    "AED": 3.6725,
```

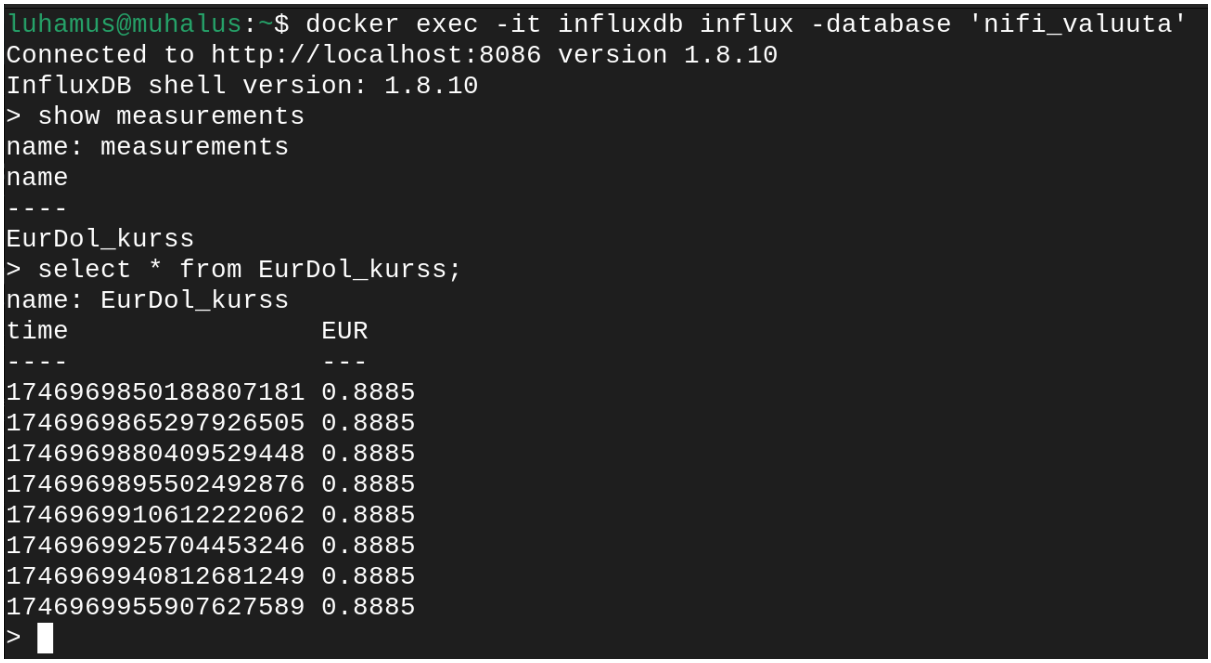
Joonis 21. ExchangeRate API andmekonveieri genereerimise jaoks jooksutati tööriista interaktiivses moodis. Siin on toodud lõik tööriista kasutamise algusest



Joonis 23. Pärast ExchangeRate API konveieri jooksutamist paigaldas tööriist selle ka Nifi platvormile



Joonis 22. ExchangeRate API andmekonveieri genereerimise jaoks jooksutati tööriista interaktiivses moodis. Siin on toodud lõik tööriista kasutamise algusest



Joonis 24. Pärast ExchangeRate API konveieri käivitamist kontrolliti ka, kas andmed jõuavad andmebaasi.

Lisa 4. Arenduseks kasutatud API-de vastuste kujud

Peatükis 3.1 “Andmed” toodi välja kaks API-t mida arenduseks kasutati. Open-Meteo poolt pakutav tasuta ilmajaama API ning Cumulocity platvormil olev API Tartu Ülikooli Delta hoone päikesepaneelide andmetega. Siin on välja toodud mõlema API vastuste kujud.


```
{
  "latitude": 58.377098,
  "longitude": 26.718277,
  "generationtime_ms": 0.04661083221435547,
  "utc_offset_seconds": 0,
  "timezone": "GMT",
  "timezone_abbreviation": "GMT",
  "elevation": 48.0,
  "current_weather_units": {
    "time": "iso8601",
    "interval": "seconds",
    "temperature": "°C",
    "windspeed": "km/h",
    "winddirection": "°",
    "is_day": "",
    "weathercode": "wmo code"
  },
  "current_weather": {
    "time": "2025-05-13T15:30",
    "interval": 900,
    "temperature": 18.5,
    "windspeed": 5.8,
    "winddirection": 311,
    "is_day": 1,
    "weathercode": 1
  }
}
```


Joonis 25. Open-Meteo API vastus, kui teha päring järgmise URL-i pihta: https://api.open-meteo.com/v1/forecast?latitude=58.38&longitude=26.72&current_weather=true

```
{
  "next": "https://t10162.iot.cs.ut.ee/measurement/measurements?dateTo=2024-
10-16T00:00:01Z&pageSize=200&source=780&?dateFrom=2024-10-
15T00:00:00Z&type=KogEN&currentPage=2",
  "self": "https://t10162.iot.cs.ut.ee/measurement/measurements?dateTo=2024-
10-16T00:00:01Z&pageSize=200&source=780&?dateFrom=2024-10-
15T00:00:00Z&type=KogEN&currentPage=1",
  "statistics": {
    "currentPage": 1,
    "pageSize": 200
  },
  "measurements": [
    {
      "self": "https://t10162.iot.cs.ut.ee/measurement/measurements/118954330",
      "time": "2022-12-23T09:17:41.000Z",
      "id": "118954330",
      "source": {
        "self": "https://t10162.iot.cs.ut.ee/inventory/managedObjects/780",
        "id": "780"
      },
      "type": "KogEN",
      "KogEN": {
        "T": {
          "unit": "kilowattHours",
          "value": 563058.8125
        }
      }
    },
    {
      "self": "https://t10162.iot.cs.ut.ee/measurement/measurements/118954330",
      "time": "2022-12-23T09:17:41.000Z",
      "id": "118954330",
      "source": {
        "self": "https://t10162.iot.cs.ut.ee/inventory/managedObjects/780",
        "id": "780"
      },
      "type": "KogEN",
      "KogEN": {
        "T": {
          "unit": "kilowattHours",
          "value": 563058.8125
        }
      }
    },...
```


Joonis 26. Päikesepaneelide API vastus, kui teha päring järgmise URL-i pihta: https://delta.iot.cs.ut.ee/measurement/measurements?source=780&dateFrom=2024-10-15T00:00:00Z&dateTo=2024-10-16T00:00:01Z&pageSize=200&type=KogEN

Lisa 5. Andmekonveierite genereerimise tööriista kood

Valminud lahenduse lähtekood on saadaval Githubis aadressil: https://github.com/Luhamus/Andmekonveierite_generaator

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Rasmus Luha**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **“Andmekonveierite ehitamise ja paigaldamise automatiseerimine”**, mille juhendajaks on **Pelle Jakovits**, reprodutseerimiseks eesmärgiga seda säiliada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost jakasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Rasmus Luha

15.05.2025