TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Jaan Nigul

Konteineripõhise keskkonna loomine TÜ kliimateenuste portaali jaoks

Bakalaureusetöö (9 EAP)

Juhendaja: Andres Luhamaa, MSc

Tartu 2025

Konteineripõhise keskkonna loomine TÜ kliimateenuste portaali jaoks

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli luua konteineripõhine platvorm erinevate kliimaandmete jagamiseks ja visualiseerimiseks, mis võimaldab luua konfigureeritavaid ja interaktiivseid vaateid, mis sobivad nii spetsiifilisteks rakendusteks kui ka prototüüpimiseks. Töö käigus loodi skaleeritav konteineripõhine platvorm kliimaandmete visualiseerimiseks, mis võimaldab käivitada isikupärastatud töölaudu, hallata kasutajaid ja sessioone ning kasutada andmeid nii lokaalselt kui ka pilves. Süsteem sisaldab autentimist, nii kasutajate kui ka konteinerite haldamist, logimist ja monitoorimist ning põhineb AWS-i teenustel ja tarkvararaamistikul Panel.

Võtmesõnad:

konteineripõhine veebiteenus, Python, Amazon Web Services, Holoviz Panel

CERCS: P175 Informaatika, süsteemiteooria; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; P500 Geofüüsika, füüsikaline okeanograafia, meteoroloogia

Development of a container-based environment for UT climate services portal

Abstract:

The aim of this bachelor's thesis was to develop a container-based platform for sharing and visualizing various climate datasets, enabling the creation of configurable and interactive dashboards suitable for both specific applications and rapid prototyping. As a result, a scalable system was built that allows launching personalized dashboards, managing users and sessions, and utilizing data stored both locally and in the cloud. The system includes user authentication, container and session management, logging and monitoring functionalities, and is built on Amazon Web Services and Panel framework.

Keywords:

container-based web application, Python, Amazon Web Services, Holoviz Panel

CERCS: P175 Informatics, systems theory; P170 Computer science, numerical analysis, systems, control; P500 Geophysics, physical oceanography, meteorology

Sisukord

Sissejuhatus

Vaatamata sellele, et keskkonnaandmeid vahendavaid portaale on mitmeid, on paljude lõppkasutajate jaoks andmetele juurdepääs piiratud keerukate tehniliste lahenduste, andmestruktuuride, suure hulga erinevate andmeprodukte ja vähese kasutajasõbralikkuse tõttu.

Bakalaureusetöö (edaspidi Atlas) eesmärk on luua platform jagamaks infot kliimaandmete kohta erinevates ajalistes ja ruumilistes mastaapides ning erinevates visualiseerimise tehnikates (lühitekstid, tabelid, graafikud, kaardid), mis võivad olla omavahel interaktiivselt seotud ja mis on koondatud eraldiseisvateks töölaudadeks (ingl *dashboard*). Atlase loomise peamine eesmärk on kasutamise lihtsus ning konfigureeritavus: iga probleemi jaoks on võimalik luua eraldi lihtne vaade või töölaud, mis just antud probleemi kõige paremini visualiseerib. Atlast võiks hakata kasutama nii spetsiifilise lõpp-produktina kui kiire prototüüpimise vahendina, mis aitaks kaasa keerukama lõpplahenduse väljatöötamisel. Kuna ei ole võimalik ette teada, milliseid andmeid täpselt on vaja millisele huvigrupile kuvada näiteks poole aasta pärast, siis peab olema võimalik luua vastav konfiguratsioon vähese vaevaga. Atlas ei ole esialgu mõeldud teenindama avalikkust, vaid üksnes spetsiifiliste vajadustega piiratud hulka kasutajaid.

Töö põhisisu on jaotatud viieks peatükiks. Esimeses peatükis kirjeldatakse töö aluseks olevat probleemi, olemasolevate lahenduste puudujääke ja lõputöö eesmärki. Lisaks tehakse lihtne ülevaade Atlase üldisest töövoost, kuidas loodav veebiteenus töötab. Teises peatükis kirjeldatakse Atlase funktsionaalsed ja mittefunktsionaalsed nõuded ning kasutajalood. Kolmandas peatükis on ülevaade peamistest lõputööga seotud tehnoloogiatest. Neljas peatükk koosneb loodud veebiteenuse ja selle komponentide detailsemast kirjeldusest. Viies peatükk sisaldab Atlase edasiarenduse võimalusi.

Töös on kasutatud juturoboti ChatGPT 4o[1] abi programmi lähtekoodi kirjutamisel, silumisel, kommenteerimisel ning töö struktuuri täpsustamisel.

[1] ChatGPT: https://chatgpt.com/

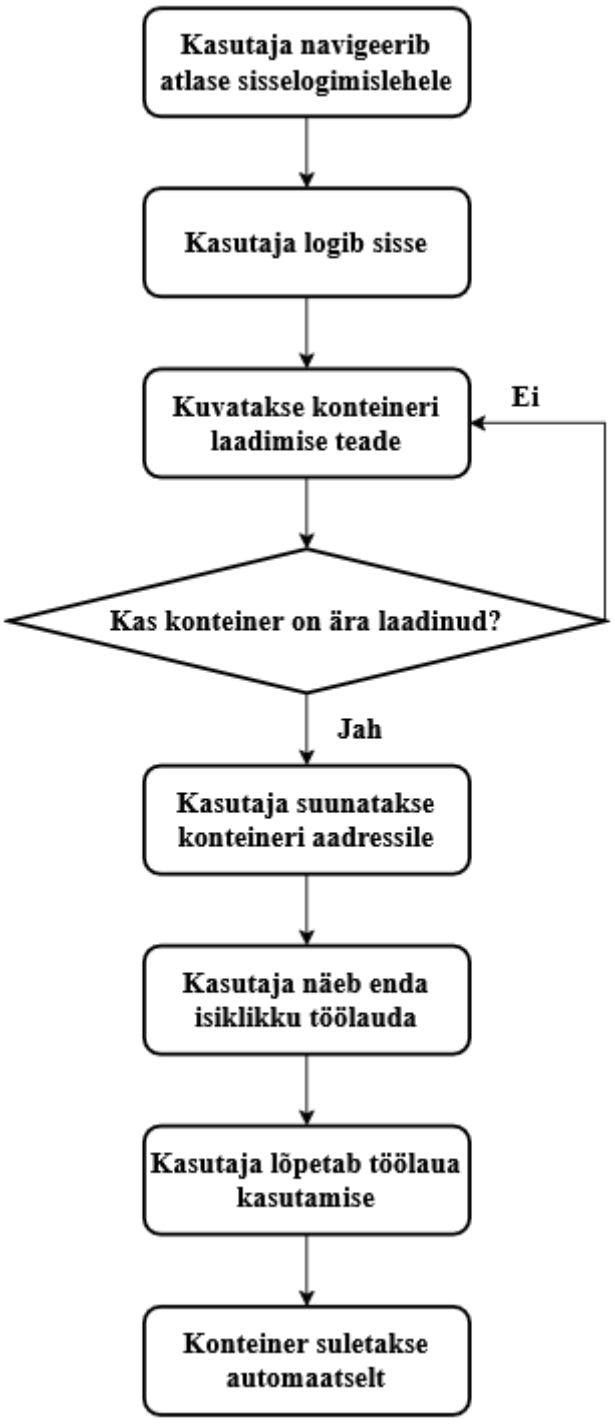
1. Probleemi ja lahenduse taust

Tartu Ülikooli kliimauuringute keskusel on jooksvalt käsil mitmeid rakendusuuringute projekte, kus analüüsitakse Eesti kohta käivaid kliimaandmeid. Andmed võivad olla seotud erinevate valdkondadega, näiteks põllumajandusega, linnalise kliima ja tuuleenergiaga. Keskkonnaandmete jagamise portaale ja keskkondi on mitmeid, näiteks Euroopa Liidu programmi Copernicus[2] teenused, sealhulgas kliimateenus C3S, atmosfääri teenus CAMS, maa monitoorimise teenuse CMLS. Samuti pakuvad keskkonnaandmeid Ameerika Ühendriikide asutused NASA[3] ja NOAA[4], lisaks on olemas erinevad pilvepõhised ligipääsuvõimalused nagu Amazon Web Services (AWS)[5] ja Google Colab[6]. Praktikas on andmetele ligipääs ja nende kasutamine aga väga piiratud. Keerukad API-liidesed, killustunud teenuste struktuur, versioonikonfliktid ja sageli puudulik dokumentatsioon raskendavad nii algajate kui ka kogenumate kasutajate tööd. Kliimauuringute keskuse kõik üksikud projektid on liialt väikesed, et nendesse kaasata eraldiseisvat veebiarendust ja arendajat andmete paremaks jagamiseks. Samas annaks interaktiivne andmete visualiseerimine kliendile palju parema ülevaate olemasolevatest andmete kasutamise võimalustest. Andmete analüüsimine toimub reeglina keskkonnas Jupyter Notebook[7], kasutades selleks programmeerimiskeele Python[8] vahendeid, kuid lõpptulemus on kas staatilised andmefailid või staatiline tekstidokument. Samas on Python-põhised graafikud lihtsasti muudetavad üle võrgu jagatavateks töölaudadeks.

Kuigi kliimauuringute keskus saab vajadusel jagada keskkonna Jupyter Notebook lähtekoodi, kaasneb sellega sageli täiendav koormus kasutajale, isegi kui ta oskab seda kasutada. Tekkida võivad probleemid, näiteks arvutis sobiva keskkonna puudumine, andmetele juurdepääsu puudumine ning puudulik kogemus konkreetse andmestiku või visualiseerimisvahendiga. Hetkel on olemas mitmeid olemasolevaid teenuseid, kus jooksutada välisel serveril Jupyter Notebook keskkondi, näiteks JupyterHub[9] ja AWS SageMaker[10] Notebooks. Samuti eksisteerib konteinerite käivitussüsteeme, millest bakalaureusetöös loodavaga on kõige sarnasem Gitpod[11]. Kõikide selles lõigus eelnevalt nimetatud keskkondade kasutamine eeldab

[2] Lisainfo Copernicus ja selle teenuste kohta: https://www.copernicus.eu/en
[3] National Aeronautics and Space Administration: https://www.nasa.gov/
[4] National Oceanic and Atmospheric Administration: https://www.noaa.gov/
[5] Amazon Web Services: https://aws.amazon.com/
[6] Google Colab: https://colab.research.google.com/
[7] Jupyter Notebook: https://jupyter.org
[8] Python: https://www.python.org/
[9] JupyterHub: https://jupyter.org/hub
[10] AWS SageMaker Notebooks: https://aws.amazon.com/sagemaker-ai/notebooks/
[11] Gitpod: https://www.gitpod.io/

kõrgemat tehnilist pädevust ning need on suunatud pigem tarkvaraarendajatele või andmeteadlasele. Sellest tulenevalt ei ole need keskkonnad sobilikud näiteks rakendusuuringutes osalevatele lõppkasutajatele, kelle eesmärk ei ole tarkvaraarendus, vaid kliimaandmete uurimine ja tõlgendamine. Atlase eesmärk on pakkuda lihtsamat ja kasutajasõbralikumat lahendust, mis võimaldab ligipääsu konteineripõhistele visualiseerimiskeskkondadele ilma tehnilise taustateadmisteta.



Joonis 1. Süsteemi üldine töövoog

Konteineripõhise visualiseerimisplatvormi üldine tööpõhimõte on üles ehitatud nii, et lõppkasutaja ei pea kokku puutuma keeruka tehnilise seadistamisega. Süsteemi üldist töövoogu on näha joonisel 1. Esmalt kasutaja navigeerib Atlase sisselogimislehele ning logib ennast sisse. Kui kasutaja on autenditud, kuvatakse talle teade, et konteinerit käivitatakse. Kui konteiner on ära laadinud ning valmis, suunatakse kasutaja sinna konteinerisse, kus ta näeb oma töölauda, mis on instituudi töötaja poolt vastavalt kasutaja profiilile ja vajadustele konfigureeritud. Töölaual saab kasutaja vaadata interaktiivseid graafikuid, valida parameetreid ja suumida kaarti sisse-välja. Kui töölaualt lahkutakse, suletakse vastav konteiner automaatselt.

2. Arendusnõuded

Selles peatükis on koondatud kliimauuringute keskuse poolt esitatud funktsionaalseid ja mittefunktsionaalseid nõudeid loodavale platvormile. Nõuded on kirja pandud sellisel kujul, nagu need arenduse varases etapis sõnastati. Need ei ole käsitletavad range määratlusena, vaid loovad esialgse arusaama süsteemi oodatavast toimimisest.

2.1 Funktsionaalsed nõuded

Järgnevalt on punktidena välja toodud rakenduse funktsionaalsed nõuded.

1. Üksik töölaud peab olema konfigureeritav puhtalt Python programmeerimiskeele vahenditega. See tähendab seda, et kõik vajalikud seadistused ja funktsioonid saab luua ja muuta otse kasutades programmeerimiskeelt Python, ilma täiendavate kasutajapoolsete raamistiketa.
2. Töölaud peab olema teostatav eraldiseisva Docker[12] konteinerina.
3. Kasutajakeskkond peab olema sõltumatu, st seda sama lehekülge korduvalt avades ei tohi vaated üksteist segada.
4. Kasutajakeskkond on olekuvaba (ingl *stateless*), st valikud, nagu parameetrid või ajavahemik, salvestuvad vaid kasutaja sessiooni jooksul ega talletu serveri mälus.

Tagamaks Atlase kasutatavust, on vajalik pidevalt monitoorida süsteemi töökorda ja jõudlust. Monitoorimine võib olla teostatud Amazon Web Services (AWS) vahenditega. Nõutavad funktsioonid monitoorimisel on järgmised:

1. Logidest ülevaate tegemine annab infot igasuguste veateadete kohta, mis on konteineri käivitamisel või kasutamise jooksul tekkinud.
2. Teenusesse sisse loginud kasutajate arvu kuvamine.
3. Lehekülje laadimise aja kuvamine, sh graafikute laadimine. Kui lehekülg laeb liiga kaua, võib jääda mulje mittetöötavast teenusest.
4. AWS Elastic Container Service[13] (ECS) kasutamisaja statistika.
5. Rakenduse jooksutamisel tekkinud rahaline kulu.

[12] Docker: https://www.docker.com
[13] AWS Elastic Container Service: https://aws.amazon.com/ecs

2.2 Mittefunktsionaalsed nõuded

Järgnevalt on punktidena välja toodud rakenduse paigalduse nõuded:

- Rakendus on kasutatav ja testitav eraldi komponentidena (töölaud, sisselogimise leht jne).
- Rakendus on käivitatav nii arendaja arvutis kui ka AWS ECS keskkonnas.
- Rakendusega seotud töölaudu ei hoita pidevalt töös, vaid vastavat töölauda kuvav Dockeri konteiner käivitatakse alles vastava keskkonna aktiveerimisel.
- Teenus peab vaikimisi olekus olema odav, st ressursside kasutus peab olema minimaalne.

2.3 Kasutajalood

Selles peatükis on kirjeldatud kasutajalood nii Atlase tavapärase kasutaja kui ka süsteemiadministraatori vaatepunktist, kes haldab kasutajaid ja nendega seotud atribuute ja konteinereid.

- Kasutaja: KOV ametnik

KOV ametnikul on vaja vaadata sademete kaarte olemasolevates kliimatingimustes ning prognoose selle kohta, kuidas need võiksid tulevikus muutuda. Olemasolevad andmed on esitatud andmekihtidena radari piltidest, jaamade mõõtmisandmetest ja mudeli järelanalüüsidest. Tuleviku andmed pärinevad mudelarvutustest ja on esitatud kaardikihtidena. Vaadata saab nii üksikuid mudeleid, ansambli statistikuid kui ka erinevust mineviku andmetest. Tehnilise poole pealt näeb see välja järgmiselt. Liikudes avalehele, näeb kasutaja sisselogimisakent. Parooli sisestamise järel näeb kasutaja laadimislehte, mis kuvab vastavat keskkonda pakkuva teenuse olekut. Kui konteiner on ära laadinud, saab kasutaja vaadata interaktiivseid graafikuid, valida parameetreid ja suumida kaarti sisse-välja.

- Kasutaja: Süsteemiadministraator

Administraatori eesmärgiks on tagada, et:

- kõik kasutajad saavad ligi just endale määratud konteinerile;
- konteinerite käivitamiseks on olemas piisavad ressursid.

Liikudes Amazon Web Services konsooli ning sealt edasi Amazon Cognito lehele, saab kasutaja lisada uusi kasutajaid ja muuta olemasolevaid. Kasutajakontoga seotud

isikupärastatud atribuute, näiteks konteineripaketi aadress ja organisatsioon, saab lisada ja muuta peale kasutajakonto loomist.

Uue töölaua loomiseks loob kasutaja Docker konteineripaketi, mis sisaldab kõik vajalikud failid konteinerisiseseks kasutaja sessioonihalduseks. Seejärel laeb kasutaja selle konteineripaketi üles teenusesse Elastic Container Registry, kust on hiljem võimalik seda kätte saada.

3. Tehnoloogiate analüüs

Selles peatükis on välja toodud platvormi loomiseks valitud tehnoloogilised tööriistad. Valikute tegemisel lähtuti arendusnõuetest.

3.1 Visualiseerimise tööriistad ja raamistikud

Kliimaandmete kujutamiseks töölaual oli oluline valida võimalikult palju selleks omavahel ühilduvaid tööriistu, et kasutada lõpptoodangu arendamisele suunatud aega maksimaalselt ning vähendada ajakulu erinevate tarkvarade mitteühilduvuse põhjuste uurimise ning parandamise peale. Üks selles töös peamiselt kasutatavaid tööriistakogumeid on HoloViz. Sophia Yangi jt artiklile [1] tuginedes on HoloViz Pythonile loodud andmete visualiseerimise teekide kollektsioon, mis aitab kasutajatel ehitada töölaudu ja nende visuaalseid komponente, kasutades lihtsat Pythoni koodi. Nad mainisid, et see tarkvarateek võimaldab valida mitme plottimisraamistiku rakendusliidese vahel, mille hulgas on näiteks Bokeh, Matplotlib ja Plotly. Autorid rõhutavad, et HoloVizi eelis üle teiste sarnaste tööriistakogumite on selle võimalused rakendust käitada palju rohkemates keskkondades, näiteks programmis Jupyter Notebook, Pythoni failis, genereeritud raportites ning pildi- ja dokumendifailides, eraldiseisvas serveris või staatilise HTML-failina veebilehel.

Tööriistakogumi HoloViz hulka kuulub ka Panel, mis on avatud lähtekoodiga tarkvarateek keerukate rakenduste ja töölaudade arendamiseks programmeerimiskeeles Python [2]. Panel toetab erinevaid andmete visualiseerimise teeke, näiteks hvPlot, HoloViews, GeoViews jt [3]. Kuna Atlase arendamiseks kasutatav programmeerimiskeel on Python, ei nõua Paneli kasutamine teiste kasutajapoolsete (ingl *frontend*) tehnoloogiate ja raamistike, näiteks React, Vue jt, kasutamist, mis muudab arendusprotsessi lihtsamaks, kuna programmeeritakse ainult ühes programmeerimiskeeles, seega puudub vajadus andmeteisenduse järele konteksti vahetusel.

Raamistikku Panel kasutades saab valida erinevate näidiste, paigutuste, vidinate, indikaatorite ja paneelide vahel, mis võimaldab kiiresti luua kohaldatud töölaudu vastavalt vajadusele. See sisaldab laia valikut visuaalseid ja funktsionaalseid komponente, sh graafikud, nupud, liugurid ja HoloViewsi[14] objektid, mida kasutatakse andmete visualiseerimiseks ka kaartidel. Komponente saab lihtsalt siduda erinevate paigutuste ja stiilidega [2]. Need omadused teevad

[14] Holoviews: https://holoviews.org

Atlase töölaudade kujundamise ja rakendamise erinevate kliimaandmete ja kasutajagruppide järgi lihtsaks ja ajasäästlikuks.

3.2 Andmete kirjeldus

Järgnevalt käsitletakse andmefailide formaate, mis on Atlase arendamisel kõige olulisemad. Pilvepõhiselt optimeeritud andmestruktuurid, nagu näiteks Zarr, GeoTIFF ja Geoparquet, võimaldavad luua töölaudu, mis suudavad filtreerida ja visualiseerida väga suuri andmehulkasid ilma, et kogu andmestikku tuleks korraga mällu laadida. Lõppkasutaja jaoks mugav andmemaht ei ületa reeglina mõnda gigabaiti, samas kui kliimaandmete suurus võib sõltuvalt probleemist ulatuda kümnetesse terabaitidesse. Selliste andmemahtude haldamine nii kasutaja arvutis või asutuse võrgukettal eeldab sageli täiendavat tehnilist tuge ning ei ole kasutajale mugav. Atlas on loodud töötama pilves paiknevate andmetega ning kasutama pilvesõbralikke andmeformaate, mille abil on võimalik käsitleda ka väga suuri andmestikke samal viisil kui väiksemaid. Lisaks kliimauuringute keskuse AWS-is hallatavatele pilveoptimiseeritud andmetele on AWS-i avalikus andmeregistris[15] suur hulk vabalt kättesaadavaid andmeid, mida saab Atlases kasutada.

3.2.1 Geoparquet

Geoparquet on andmesalvestuse formaat, mis põhineb Apache Parquet'il, kuid on kohandatud georuumiliste andmete jaoks [4, 5]. See lisab georuumilisi tüüpe, näiteks punkt, joon ja hulknurk [5]. Apache Parquet'i kodulehel [5] kirjeldatakse Parquet'i kui avatud lähtekoodiga veerupõhist andmefailiformaati, mis on loodud tõhusaks andmete salvestamiseks ja päringute tegemiseks. Samuti märgitakse, et Apache Parquet võimaldab suure jõudlusega andmete tihendamist ja kodeerimist, mis on kasulik suurte ja keerukate andmekogumite töötlemisel. Lisaks on kodulehel öeldud, et Parquet'i toetavad mitmed programmeerimiskeeled ja analüüsitööriistad, muutes selle laialdaselt kasutatavaks erinevates andmeanalüüsi keskkondades. Geoparquet ja Parquet faile saab lugeda vastavalt tarkvarateekide GeoPandas ja Pandas abil [6, 7].

3.2.2 GeoTIFF

GeoTIFFi spetsifikatsioon sisaldab TIFF-failidele mõeldud märgiste (ingl *tags*) kogumit, mis on loodud geograafilise ja kartograafilise teabe kirjeldamiseks [8]. Selline teave võib olla seotud rasterpiltidega, mis on saadud satelliidipiltidelt, skaneeritud aerofotodest, kaartidest,

[15] Registry of Open Data on AWS: https://registry.opendata.aws/

digitaalse kõrgusmudeli andmetest või geograafilise analüüsi tulemusena [8]. GeoTIFF võimaldab siduda rasterpildi tuntud koordinaatsüsteemi või kaardiprojektsiooniga ja neid analüüsida [8].

Pilveoptimeeritud GeoTIFF (*Cloud Optimized GeoTIFF*, COG) on GeoTIFFi erivorming, mis on loodud tõhusaks kasutamiseks pilvekeskkondades [9]. COG-failid on struktureeritud nii, et hüperteksti edastuse protokoll (ingl *Hypertext Transfer Protocol*, HTTP) GET vahemikupäringute abil on võimalik küsida ainult vajalikke failiosi ilma tervet faili allalaadimata [9]. GeoTIFF faile, olenemata pilveoptimeeringu kasutamisest, saab lugeda samal moel, näiteks tarkvarateekide Rasterio ja Xarray abil [9, 10, 11].

3.2.3 Zarr

Veebipõhise juhendi *Cloud-Optimized Geospatial Formats Guide* [12] kohaselt on Zarr failiformaat välja töötatud väga suurte n-mõõtmeliste andmehulkade salvestamiseks ja töötlemiseks, mis ületavad tüüpiliste kasutajaseadmete töömälu mahutavuse. Allika andmetel võimaldab Zarr andmete tõhusat tihendamist ja struktureerimist viisil, mis toetab selektiivset andmepöördumist – kasutajal on võimalik pärida üksnes vajalikud andmeplokid (ingl *chunks*) ning hajutada andmetöötluskoormus mitmete arvutusressursside vahel. Juhendis on kirjas, et Zarr-andmekogum koosneb tihendatud ja plokkideks jaotatud n-mõõtmelistest massiividest, kusjuures iga andmeplokk esindab võrdses mahus osa kogu massiivist ning talletatakse eraldi failina. Selline arhitektuur võimaldab lokaalseid jaotatud lugemis- ja kirjutamistoiminguid ilma vajaduseta kogu andmestikku mällu laadida. Selle tõttu sobib Zarr hästi paralleelseks töötlemiseks ja kasutamiseks pilvepõhistes objektipõhistes salvestuskeskkondades. Üheks Zarr-formaadi oluliseks eeliseks võrreldes näiteks andmeformaatidega GeoTIFF või NetCDF[16] on selle täielik paralleelsus nii lugemis- kui ka kirjutamisoperatsioonides [13]. See tuleneb sellest, et iga andmeplokk on eraldiseisev fail. Allikas *Cloud-Optimized Geospatial Formats Guide* [12] on ka kirjas, et selle failiformaadi viilutamiseks ja osade välja valimiseks saab kasutada tarkvarateeki Xarray.

3.3 Kasutajate ligipääs ja autentimine

Selles peatükis tuuakse ülevaade kasutatavatest tehnoloogiatest kasutajate ligipääsul ja autentimisel. Kuna Atlas on mõeldud esialgu ainult teenindama piiratud hulka kasutajaid,

[16] NetCDF: https://www.unidata.ucar.edu/software/netcdf/

toimub kasutajakonto registreerimine manuaalselt veebiteenuse haldaja poolt. Selle tõttu ei sisalda Atlase sisselogimisleht kasutaja registreerimise funktsionaalsust.

3.3.1 Amazon Cognito

Amazon Cognito on veebi- ja mobiilirakenduste identiteediplatvorm. See täidab samaaegselt kasutajakataloogi, autentimisserveri ning autoriseerimisteenuse rolli nii OAuth 2.0[17] pääsutooken (ingl *access token*) kui ka AWS-i mandaadi (ingl *credentials*) haldamisel [14]. Selle teenuse kasutamine aitab aega kokku hoida, mis muidu kuluks kasutajate paroolide haldamise, autentimise ja muu vajaliku taristu arendamise peale.

Cognitos saab salvestada kasutajapõhiseid kohandatud atribuute, mille abil on võimalik seostada kasutaja tema organisatsiooniga ja määratud konteineripaketi aadressiga [15]. Seetõttu ei ole vaja täiendavat andmebaasi kasutajate andmete säilitamiseks, sest Cognito vastutab nii kasutajate registreerimise, autentimise kui ka nende andmete haldamise eest. Cognito ei võimalda kohandatud atribuutide põhjal teha päringuid samal viisil nagu relatsioonilises andmebaasis [16]. Näiteks ei saa sellist päringut esitada, kus tagastatakse kõik kasutajad, kellele on määratud kindel konteineripakett. Kuna Atlas on esialgu mõeldud vaid piiratud hulka kasutajaid teenindama, ei ole suureks takistuseks kasutajate kogumist otsitava atribuudi käsitsi leidmine.

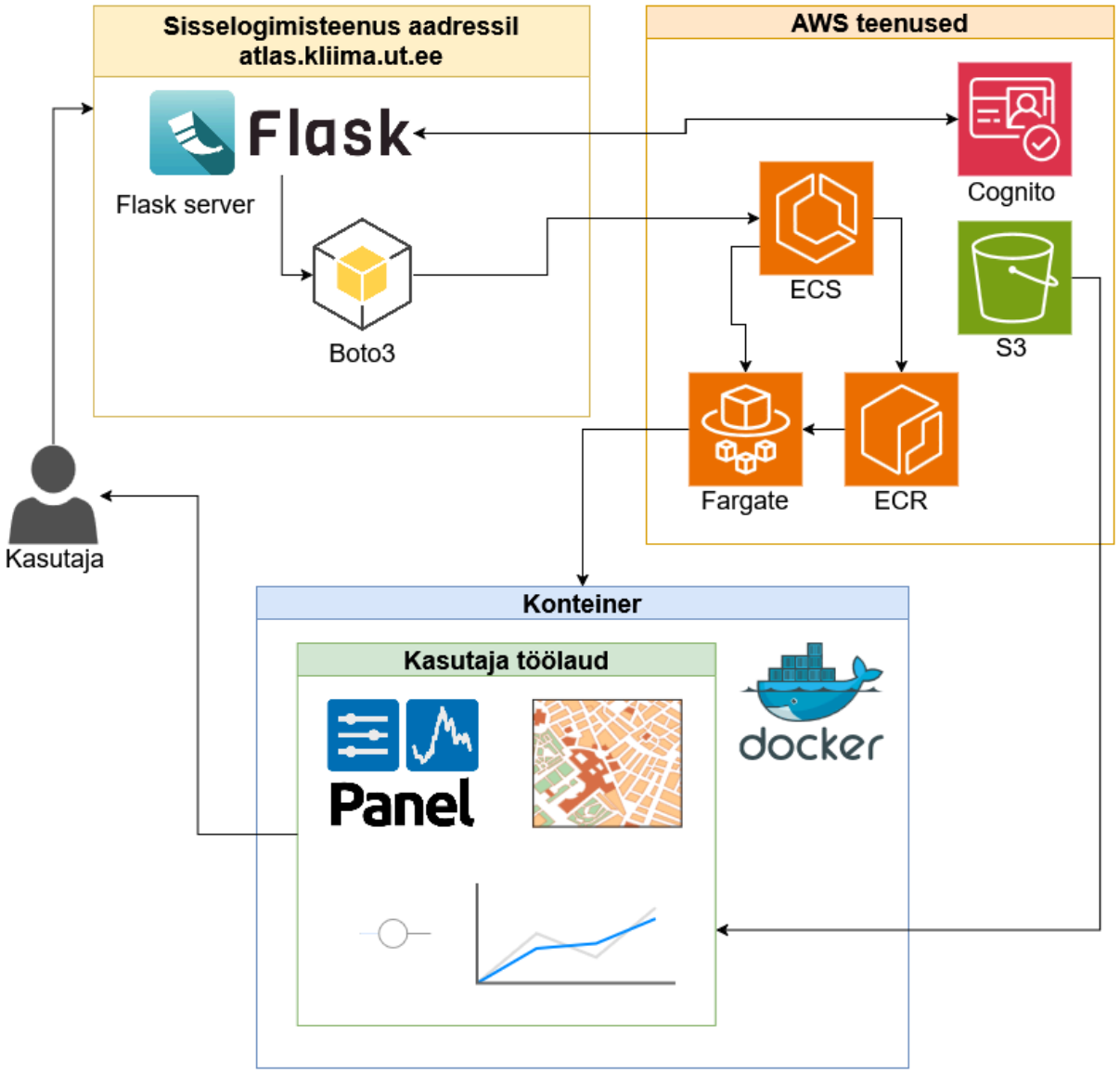
3.3.2 Amazon Elastic Container Service

Amazon Elastic Container Service on AWS-i konteinerite orkestreerimise teenus, mis lihtsustab konteineriseeritud rakenduste käitamist, haldamist ja skaleeritavust. See ühildub AWS-i teiste teenustega, näiteks konteineriregistriga Elastic Container Registry (ECR), ning ka kolmanda osapoolte teenustega, näiteks Docker [17]. Atlase jaoks on ECS-i põhilised mõisted tegumi definitsioon (ingl *Task Definition*), klaster (ingl *Cluster*) ja tegum (ingl *Task*). Tegumi definitsioon on JSON-formaadis mall, mis määrab ära, kuidas konteinerit käivitada. See sisaldab teavet kasutatava konteineripaketi kohta (näiteks ECR-i või Docker Hubi aadress), protsessori ja mälu ressursside seadistused, võrguseaded ning infrastruktuuri kohta, millel tegumeid käitatakse [18]. Klaster on loogiline kogum ressursse, mille sees tegumid töötavad. Tegum on tegumidefinitsiooni põhjal käivitatud tööprotsess konteineris [19].

[17] OAuth 2.0: https://oauth.net/2/

4. Loodud veebiteenus

Järgnevas peatükis kirjeldatakse lõputöö käigus valminud veebipõhist süsteemi, mille eesmärk on võimaldada autenditud kasutajatel ligipääsu personaalsele töölauale. Süsteem koosneb mitmest komponendist, milleks on sisselogimisteenus, konteineripõhine töölaud, kasutajahaldus ja logimisfunktsioonid. Link programmi lähtekoodile asub lisas I .



Joonis 2. Atlase üldine arhitektuur

Loodud veebiteenuse üldist arhitektuuri illustreerib joonis 2. Kasutaja pöördub esmalt sisselogimisteenuse poole, mis on loodud raamistikus Flask. Autentimine toimub Amazon Cognito vahendusel. Pärast edukat autentimist käivitab sisselogimisteenus konteineri, kasutades AWS pilveteenuseid, eelkõige ECS koos Fargate töörežiimiga, mis võimaldab hallata konteinereid ilma virtuaalmasinaid otseselt seadistamata. Kasutaja töölaud töötab

isoleeritud Docker-konteineris, mida kasutav konteineripakett on alla laetud konteineriregistrist ECR. Töölaud on loodud tarkvararaamistiku Panel abil. Andmeid saab laadida konteineri seest, kuid ka välisest salvestusruumist, näiteks objektsalvestusteenusest Amazon S3[18].

4.1 Kasutaja autentimine

Kasutajaandmed hoitakse identiteediplatvormis Amazon Cognito, kus igale kasutajale on määratud neli atribuuti: `sub`, `email`, `custom:organization` ja `custom:containerImage`. Atribuut sub on kasutaja püsiv unikaalne identifikaator, mida kasutatakse süsteemis kasutaja üheselt tuvastamiseks, `email` kirjeldab kasutaja e-posti aadressi, mis on vajalik sisselogimisel. Väli `custom:organization` on kasutaja organisatsiooni nimi ja `custom:containerImage` tähistab kasutajale määratud konteineripaketi aadressi, mis käivitatakse Atlasesse sisselogimisel.

Loodud veebirakenduses kasutatakse Pythonil põhinevat veebiraamistikku Flask[19], mille abil hallatakse kasutajaliidese meetodeid ja sessiooniandmeid. Kasutaja autentimine toimub OpenID Connect (OIDC) protokolli kaudu, kasutades Amazon Cognito identiteedihaldusteenust ning OAuth 2.0 voogu Authlib teegi toel. Sisselogimisloogika on realiseeritud eraldi failis „login.py“, kus asuvad kõik Cognito autentimisega seotud marsruudid ja sessioonikäsitlused.

Sisselogimise protsess algab, kui kasutaja navigeerib `/login` marsruudile. Seal käivitab Authlib autoriseerimisvoo, suunates kasutaja AWS Cognito sisselogimislehele. Peale edukat autentimist suunatakse kasutaja tagasi rakenduse `/authorize` marsruudile koos autoriseerimiskoodiga, mille alusel vahetatakse välja tooken nimega `id_token` ja hangitakse kasutajainfo. Saadud kasutajainfo, sealhulgas kasutaja `sub`, e-posti aadress ning kohandatud atribuudid (nt konteineripaketi aadress), salvestatakse Flaski sessiooniobjekti.

Autentimise järgselt suunatakse kasutaja `/dashboard` marsruudile koos parameetriga `id_token`, kus kuvatakse konteineri käivitamise olekut marsruudi `/task_status` kaudu, mis tagastab konteineri oleku reaalajas (`PROVISIONING`, `PENDING`, `RUNNING`, `STOPPED`). Selle vaate JavaScripti kaudu tehakse taustal päring `/launch` meetodile, mis käivitab isikupärastatud konteineri AWS ECS Fargate platvormil, kasutades selleks eelnevalt

[18] Amazon S3: https://aws.amazon.com/s3/
[19] Flask: https://flask.palletsprojects.com/en/stable/

määratud konteineripaketti ja kasutaja e-posti aadressi põhjal loodud unikaalset tegumi pere (ingl *task family*).

Meetod `/launch` teostab vajaliku ECS konteineri registreerimise ja käivitamise ning ootab ära, kuni konteiner on seisundis `RUNNING` ja talle on määratud avalik IP-aadress. Kui konteiner on edukalt käivitatud, tagastab server JSON-vastuses konteineri IP, mille alusel suunatakse kasutaja automaatselt edasi tema isiklikule töölauale.

4.2 Konteinerite haldus

Iga töölaud töötab eraldi konteineris, mis käivitatakse vastavalt kasutaja autentimisel. Selline lahendus tagab skaleeritavuse ning sõltumatuse teistest kasutaja seanssidest. Selles peatükis kirjeldatakse Atlase süsteemis kasutatavat konteineripõhist arhitektuuri, sealhulgas konteinerite sisemist failistruktuuri, käivitamise- ja peatamisloogikat ning sessioonihalduse tehnilist ülesehitust.

4.2.1 Konteinerisisene struktuur

Lisaks tarkvarapaketiga Panel loodud töölaua failile tuleb konteineripõhise tööruumi loomiseks luua kindel failistruktuur, mille alusel Docker konteineripaketi (ingl *Docker image*) koostab. Atlase konteinerite jaoks loodud Docker-pakett sisaldab järgmisi olulisi faile, millest igaühel on konkreetne ülesanne konteineri tööprotsessis.

1. „Dockerfile“ määrab ära kasutatavad tarkvarapaketid faili „environment.yml“ abil ning käsud, mis jooksutatakse konteineri käivitamise ajal. Failis kasutatakse baaskujuna „condaforge/miniforge3:latest“[20], mis kasutab paketihaldurit Conda ning minimaalset Linuxi distributsiooni Ubuntu. Pärast keskkonna loomist kopeeritakse konteinerisse Atlasega seotud lähtefailid nimedega „app.py“, „verification.py“, „setup_callbacks.py“ ja „dashboard.py“. Fail „Dockerfile“ sisaldab ka Paneli rakenduse jaoks pordi 5006 avamist, Conda-keskkonna aktiveerimist ning „app.py“ aktiveerimist kindlate argumentidega. Täpsem teave sessioonihaldusega seotud käivitusargumentide kohta, sealhulgas „setup_callbacks.py“ kasutus, on esitatud alapeatükis „Sessioonihaldus“, kuna need funktsionaalsused on otseselt seotud konteineri elutsükli juhtimisega.

[20] Baaskuju „condaforge/miniforge3“: https://hub.docker.com/r/condaforge/miniforge3

2. Fail „environment.yml“ sisaldab sõltuvusi ja konfiguratsioone Conda-keskkonna loomiseks. See tagab, et töölaua toimimiseks vajalikud Pythoni teegid ja nende versioonid on konteinerisse korrektselt paigaldatud.
3. Fail „app.py“ on konteineri peamine käivitusfail, mille ülesandeks on töölauda kuvava Panel-rakenduse initsialiseerimine ning varasemalt autentimisest konteinerisse kaasa antud tookeni `id_token` verifitseerimine faili „verification.py“ abil.
4. Fail „dashboard.py“ sisaldab kogu kasutajale nähtava Panel-töölaua loogikat ning selles määratakse rakenduse visuaalne ülesehitus, kasutajaliidesed, andmete laadimine ja töötlemine ning interaktiivsete graafikute ja kaartide loomine. Kuna konteineris olev Panel-rakendus peab olema käivitatav läbi Paneli käsurea vahendusel (`panel serve app.py`), peab kogu kasutajaliidese loogika olema defineeritud funktsiooni sees, mis võtab argumentideks tookeni, mis on saadud autentimisprotsessist, ja konteineri IP-aadressi. Kasutaja väljalogimise funktsionaalsuseks tuleb sellesse faili lisada funktsioon `on_logout_click` ning lisada ka objektid `logout_pane` ja `logout_button`, mida hiljem Paneli paigutusse `layout` ka lisatakse.
5. Failis „verification.py“ asuvad vajalikud funktsioonid kasutaja verifitseerimiseks, kui luuakse ühendus konteineriga.
6. Fail „setup_callbacks.py“ sisaldab programmikoodi, mis jooksutatakse enne peamise Panel-rakenduse käitamist ning sisaldab kasutaja sessioonihalduse loogikat.

Atlase konteinerites peavad sisalduma kõik eelnevalt loetletud failid, et tagada rakenduse korrektne toimimine, sealhulgas kasutaja verifitseerimine ja sessioonihaldus. Failistruktuur on üles ehitatud selliselt, et igal failil on selgelt piiritletud funktsionaalsus, mida see rakendusele lisab. Selline ülesehitus muudab erinevate funktsionaalsete komponentide muutmise ja arendamise paindlikumaks ning lihtsustab süsteemi hooldust ja laiendamist tulevikus.

4.2.2 Konteinerite käivitamine

ECS konteinerite käivitamiseks kasutatakse Atlases AWS ametlikku tarkvaraarenduskomplekti Boto3[21], mis on mõeldud kasutamiseks Pythonis. Konteinerite käivitamisega ja peatamisega seotud funktsioonid asuvad failis „awscontroller.py“, mis on kasutuses koos sisselogimisega seotud failiga „login.py“.

[21] Boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Konteinerite käivitamise loogika asub funktsioonis `run_ecs_container_fargate`, kus esmalt kontrollitakse, kas kasutaja e-posti aadressi põhjal tuletatud tegumi pere on juba olemas. Kui ei ole, kopeeritakse olemasolev baasdefinitsioon nimega „atlas-task“ ning kohandatakse see kasutajaspetsiifiliseks, muutes konteineripaketi aadressi ja tegumipere nime. Seejärel registreeritakse uus tegumi definitsioon, kus määratakse kasutatavate ressursside hulk, näiteks protsessori tuumade arv ja mälu kogus.

Registreeritud tegum käivitatakse määratud ECS klastris Fargate töörežiimis, kasutades `awsvpc` võrgurežiimi, mis tagab igale konteinerile omaenda võrguliidese ja avaliku IP-aadressi. Lisaks määratakse konteineri käivitamisel ära konkreetsed alamvõrgud (ingl *subnets*) ja turberühmad (ingl *security groups*). Käivitatud tegumi olekut jälgitakse kuni see jõuab seisundisse `RUNNING`. Kui konteiner on aktiivne, tuvastatakse selle külge seotud võrguliidese (Elastic Network Interface, ENI) kaudu avalik IPv4-aadress, mis tagastatakse veebirakendusele JSON-vormingus. See võimaldab suunata kasutaja automaatselt vastava konteineri kasutajaliidesele.

Lisaks on loodud funktsioon `stop_ecs_container_fargate`, mille abil lõpetatakse konteineri töö kasutaja sessiooni lõppemisel või väljalogimisel. Käivitatud konteineri IP-aadress ja tegumi ARN salvestatakse ajutiselt serverisse, võimaldades neid vajadusel tuvastada ja peatada.

4.2.3 Sessioonihaldus

Atlase juures on oluline ressursside minimaalne kasutus, et vähendada nende liigkasutust, seega on vaja kontrollida iga käimasoleva konteineri tööaega, et see töötaks vaid nii kaua, kui on vajalik, kuna ECS Fargate’i eest arvestatakse tasu sekundi täpsusega. Selleks on konteineri sees rakendatud mehhanismid, mis automaatselt peatavad kasutust mitteomavaid või liiga kaua aktiivsena hoitud konteinereid. Need funktsioonid asuvad failis „setup_callbacks.py“ ning käivitatakse töölaua rakenduse laadimisel käsurea argumendi `--setup setup_callbacks.py` abil. Selles failis on võimalik jooksutada Python koodi enne peamise töölaua programmi jooksutamist, mis on oluline näiteks kasutaja ühendumise tuvastamiseks.

Paneli raamistik võimaldab tuvastada kasutajaseansi loomist ja lõppemist sellele sisseehitatud sündmuste ohjurite (ingl *handlers*) abil:

- `on_session_created` aktiveerub uue kasutajaseansi loomisel, millega märgitakse konteiner aktiivseks. Selle sündmusega on seotud funktsioon `session_created(session_context)`, mis salvestab kasutajaseansi tookeni ja konteineri IP-aadressi;
- `on_session_destroyed` käivitatakse kasutajaseansi lõppemisel või akna ootamatu sulgemise korral. Selle sündmusega kutsutakse välja funktsioon `session_destroyed(session_context)`, mis algatab konteineri sulemise protsessi.

Konteinerite ja sealhulgas ECS tegumi peatamiseks on mitu mehhanismi, mis aktiveeruvad erinevates olukordades.

Esimene selline juhtum on kasutajapoolne sihipärane väljalogimine, mille puhul vajutab kasutaja töölaual vastavale nupule. Selle tulemusena saadetakse sisselogimisserverile päring `/logout` marsruudile koos konteineri IP-aadressiga, mis algatab seotud ECS tegumi peatamise, peale mida suunatakse kasutaja tagasi sisselogimislehele. Kuna väljalogimisnupu loogika asub failis „dashboard.py“, on selle olukorra käsitlemiseks loodud funktsioon `logout(container_ip)` failis „verification.py“.

Teine võimalik olukord on kasutaja brauseri akna sulgemine ilma kasutaja poolt väljalogimisnupule vajutamata. Paneli rakenduse jooksutamise argumentide seas on sellise olukorra jaoks kaks olulist parameetrit: `--unused-session-lifetime 300000` ja `--check-unused-sessions 10000`. Esimene neist määrab, kaua tohib kasutamata seanss (ingl *unused session*) kesta, teine määrab ajaintervalli, mille järel kontrollitakse kasutamata seansse. Kui määratud ajavahemik on möödunud, käivitatakse failis „setup_callbacks.py“ funktsioon `session_destroyed(session_context)`, mis saadab esmalt sisselogimisserverile päringu väljalogimiseks sarnaselt tavalise väljalogimisega. Selle päringuga erindi tekkimisel või muu rikke korral suletakse programm käsuga `sys.exit(15)`, et kindlasti tagada konteineri sulgumine.

Kolmas potentsiaalne stsenaarium tekib juhul, kui kasutaja sulgeb brauseri akna enne, kui konteiner on jõudnud töövalmidusse ning kasutaja on suunatud edasi personaalsele töölauale. See võib juhtuda näiteks olukorras, kus kasutaja katkestab protsessi ajal, mil konteinerit alles käivitatakse ning kasutajale kuvatakse laadimisekraani. Selle olukorra haldamiseks on loodud eraldi lõim failis „setup_callbacks.py“, mis käivitab funktsiooni `kill_if_no_session_after_n_seconds`. See funktsioon võtab argumendiks aja

sekundites ning ootab selle aja, peale mida kontrollitakse, kas kasutajaseanss on loodud. Kui kasutajaseanssi pole loodud määratud aja möödumisel, kutsutakse välja funktsioon `os._exit(15)`, mis lõpetab kogu protsessi ning selle järel konteineri.

Neljandaks konteineri sulemise mehhanismiks on maksimaalse lubatud tööaja möödumine, sõltumata kasutaja aktiivsusest. See lahendus tagab, et ükski konteiner ei jää jooksma määramatuks ajaks, mis võiks põhjustada ebavajalikku ressursikulu. Failis „setup_callbacks.py“ käivitatakse selleks eraldi lõim, mis kutsub välja funktsiooni `kill_container_after_x_hours(hours)`, kus argument `hours` tähistab maksimaalset lubatud tööaega tundides. Selle aja möödudes suletakse konteiner funktsiooniga `os._exit(15)` sarnaselt kolmanda olukorraga.

Sessioonihalduse lahendamine konteineri sees võimaldab igal kasutajal iseseisvalt hallata oma töölauda ilma keskse serveripoolse jälgimiseta. Sellise lähenemise eelisteks üle keskse kontrollsüsteemi kasutamise on parem tõrkekindlus, sest kogu süsteemi toimimine ei ole haavatav üheainsa tõrke suhtes, näiteks võrgukatkestuse või muu tehnilise tõrke korral. Süsteemi keerukus ja koormus ei suurene kasutajate arvu kasvades, sest iga uus konteiner vastutab ise enda tööea eest. Teiseks eeliseks on üksiku töölaua konfigureeritavus, st sessioonihalduse sätteid saab individuaalselt muuta vastavalt vajadusele. Konteineripõhise sessioonihalduse puudusteks on selle loogika muutmine, kui seda on vaja teha näiteks kõikide konteineripakettide puhul. See eeldaks endas iga konteineripaketi muutmist ning uuesti ehitamist. Kuna Atlas on mõeldud teenindama vaid piiratud hulka kasutajaid, ei ole selline käsitlusviis hetkel probleemne. Piiratud kasutajabaasi korral on konteineripõhise sessioonihalduse halduskoormus väike ning vajadus muudatuste järele on harv.

4.3 Töölauad

Lõputöö raames ei valminud konkreetset ühetaolist töölauda, vaid loodi töölaudade üldine struktuur ja toimimispõhimõtted, mida saab kohandada vastavalt konkreetsetele kasutajate vajadustele ja andmestikele. Testimiseks kasutati kliimauuringute keskuse poolt tehtud lihtsat kliimamudelite andmete vaatamise töölauda. Atlase töölauad on loodud kasutama tarkvarateeki Panel ning seda kasutavad töölauad peavad kasutaja väljalogimise funktsionaalsuse jaoks sisaldama teatud funktsioone ja objekte, mis on kirjeldatud peatükis „Konteinerisisene struktuur“.

Paneli töölaud koosneb kasutajaliidese komponentidest, näiteks nupud, liugurid, valikud, mille kaudu kasutaja saab määrata parameetreid või vahetada vaateid. Need komponendid on

seotud andmepõhiste funktsioonidega, mis tagastavad graafilise väljundi. Väljunditeks võivad olla näiteks rasterandmetel põhinevad visualiseeringud, ajaseeriad või kaardid.

Töölaual kuvatavad andmed võivad paikneda nii konteineri failisüsteemis kui ka välises pilvesalvestuskeskkonnas, näiteks objektsalvestusteenusest Amazon S3. Kui andmed asuvad konteineri sees, peavad need olema eelnevalt konteineripaketti lisatud. See sobib väiksemamahuliste ja staatiliste andmekogumite korral. Suuremate või pidevalt uuenevate andmete puhul on mõistlikum kasutada pilvesalvestuskeskkonda, kus andmed on saadaval võrgupõhiselt ja vajadusel dünaamiliselt laaditavad. Näiteks objektsalvestusteenusest S3 on võimalik laadida ainult hetkel visualiseerimiseks vajaminevaid Zarr-formaadis andmeplokke, mis omakorda oluliselt vähendab mälukasutust, mis on kasuks piiratud mäluressurssidega konteinerite kasutamisel. Sama kehtib ka teiste pilveoptimiseeritud andmeformaatide, nagu Parquet ja GeoTIFF, kohta. Andmete laadimise tõrgete korral on võimalik näidata kasutajale veateateid vastavalt töölaua konfiguratsioonile.

Töölauale sisenedes salvestatakse konteineripoolselt esmalt autentimise protsessist kaasa saadud tooken, mille kehtivust, päritolu ning aegumist kontrollitakse AWS Cognito avaliku võtme abil. Kui kasutaja ei ole autoriseeritud, kuvatakse töölaua sisu asemel veateade. Kui autoriseeritud kasutajaga samaaegselt üritab ühendust luua ka autoriseerimata kasutaja, ei mõjuta see autoriseeritud kasutaja töölaua kasutuskogemust, kuna veateade kuvatakse üksnes autoriseerimata kasutajale.

4.4 Veebirakenduse majutus

Lõputöö käigus valminud veebiportaali sisselogimisteenus on majutatud Tartu Ülikooli kliimauuringute keskuse serveris, kasutades selle haldamiseks serverite ja veebilehtede haldamiskeskkonda cPanel[22]. Sisselogimisserveris kasutatakse turvalise andmevahetuse tagamiseks turvalise hüperteksti edastuse protokolli (ingl *Hypertext Transfer Protocol Secure*, HTTPS) ühenduse jaoks ning on loodud ka eraldi alamdomeen „atlas.kliima.ut.ee“, mille kaudu on portaal kättesaadav. Serveris paikneb ka konfiguratsioonifail „.env“, mis sisaldab autentimiseks ja konteinerite käivitamiseks vajalikke keskkonnamuutujaid.

Kõik konteinerite haldamisega seotud teenused, sealhulgas ECS, ECR ja Cognito, asuvad Tartu Ülikooli kliimauuringute keskuse hallatavas AWS kontos, millel on piiratud õigused vastavalt kasutaja ülesannetele. Sama konto kaudu on ligipääs ka kõikidele teistele Atlases kasutatavatele AWS teenustele.

[22] cPanel: https://cpanel.net/

4.5 Logimine ja monitoorimine

Lõputöös valminud veebirakenduse süsteemi logimise ja monitoorimise jaoks on kasutatud erinevaid teenuseid, sest süsteemi komponendid on hajusalt paigutatud.

Sisselogimisserveri ja konteineri käivituse informatsiooni logimisega seoses on kasutatud programmaatilist faili logimist ja ka vähesel määral serverite ja veebilehtede haldamiskeskkonda cPanel. Kuna Flask sisaldab juba sisseehitatud logimist, on lihtsuse mõttes faili „awscontroller.py“ hetkel implementeeritud eraldi funktsioon nimega log_into_file faili logimise jaoks. Seda on tehtud põhjusel, et Flaski standardlogidele ei olnud arendajal õiguste puudumise tõttu võimalik ligi pääseda, sest nad asuvad sisselogimisserverit jooksutavas masinas. Lisaks võimaldab loodud funktsioon vältida Flaski logimiskonfiguratsiooni muutmise vajadust, mis oleks ajamahukas ning lõputöö raamistikus ebapraktiline. Logifail, millesse loodud funktsioon kirjutab on nimega „app.log“ ning sisaldab näiteks kasutaja sisselogimise teatisi, ECS tegumi käivitusi ning väljalogimiste teavet. Sisselogimisserveri teenuse veateated on saadaval cPaneli kasutajaliidese kaudu, kus on võimalik piiratud mahus jälgida vaid serveri tasemel ilmnevaid veateateid.

Konteinerisiseste logide, sealhulgas Panel-rakenduse ja muude konteinerisiseste protsesside väljundi kogumiseks ja jälgimiseks kasutatakse AWS Cloudwatch[23] teenust. Cloudwatch on integreeritud ECS teenusega, mis võimaldab automaatselt suunata iga konteineri standardväljundi- ja veateated AWS pilvelogidesse. Neid logisid saab vaadata AWS konsoolist ning iga tekitatud tegumi kohta on eraldi logid, mis kustuvad automaatselt teatud aja möödumisel, kui tegum koos konteineriga on suletud.

Lisaks võimaldab AWS konsool hallata tegumeid ja nende definitsioone, jälgida tegumite olekut, analüüsida ressursikasutust ning vajadusel kiiresti tuvastada ja lahendada süsteemis esinevaid probleeme.

[23] Amazon CloudWatch: https://aws.amazon.com/cloudwatch/

5. Edasiarenduse võimalused

Atlase arendus lõputöö raames keskendus süsteemi põhikomponentide ehk autentimise, konteineripõhise töölaua, sessioonihalduse ja logimise toimivate versioonide loomisele. Kuigi arendatud lahendus täidab oma eesmärki piiratud kasutajate hulga teenimisel, on süsteemil mitmeid edasiarendusvõimalusi. Selles peatükis kirjeldatakse edasiarenduse võimalusi, mis puudutavad peamiselt Atlase turvalisust.

Hetkel kasutavad Atlase töölauad ECS serverivaba Fargate-režiimil töötavaid konteinereid, millele ligipääs toimub HTTP-protokolli kaudu. Praegune lahendus töötab selliselt, et iga konteiner omab uut IP-aadressi ning kasutaja suunatakse otse oma töölaua aadressile. Lisaks antakse konteineri ühtsele ressursilokaatorile (ingl *Uniform Resource Locator*, URL) kaasa kasutaja tooken `id_token`, mis muudab URL-i pikaks ning seab tookeni nähtavusega seotud turvariskid. Kuigi selline lahendus on tehniliselt lihtne ja piisav, arvestades et Atlase töölaudades ei töödelda tundlikke ega isikuandmeid, ei taga HTTP-protokoll turvalist andmevahetust võrgu kaudu. Kuigi transpordikihi turbeprotokolli kasutamine (ingl *Transport Layer Security*, TLS) ei ole praeguses kasutusstsenaariumis hädavajalik, oleks selle rakendamine soovitatav, et ennetada potentsiaalseid turvariske ja tõsta süsteemi usaldusväärsust, eriti kui kasutajaskond või funktsionaalsus tulevikus laieneb.

Kõige mõistlikum ja lihtsam viis TLS-i kasutamiseks on võtta kasutusele AWS teenus Application Load Balancer (ALB)[24], mis võimaldab suunata kogu liikluse läbi keskse koormusjaoturi ja lõpetada TLS-ühendus ALB tasemel ning hoida kasutaja samal domeenil sisselogimisserveriga. See tähendab, et välised ühendused Fargate konteineriteni toimuvad turvaliselt läbi HTTPS-protokolli, samal ajal kui konteineri sees võib jääda kasutusele lihtsam HTTP-ühendus. Sellise lahenduse implementeerimine Atlases nõuaks mõningaid konteinerite käivitamisloogika muudatusi. Kuna iga kasutaja konteineripakett võib olla erineva konfiguratsiooniga, ei saa kasutada ühtset teenust kõikide konteinerite jaoks. Lisaks peab ECS-i tegum olema seotud teenusega (ingl *Service*), et ALB kasutamine oleks võimalik. Seetõttu tuleks enne konteineri käivitamist luua uus teenus või kasutada olemasolevat, mis on seotud konkreetse kasutaja ja tema konteineripaketiga. AWS ALB teenuse kasutamise peamiseks puudujäägiks on teenuse püsiv lisakulu, sest selle maksumus on tunnipõhine ning seda arvestatakse ka siis, kui ükski kasutaja töölauda ei kasuta.

[24] AWS Application Load Balancer: https://aws.amazon.com/elasticloadbalancing/application-load-balancer/

Teiseks võimalikuks turvalisuse parandamise meetmeks on sissetulevate ühenduste piiramine IP-aadressi alusel. Selle abil saab kontrollida, millistest võrkudest on ligipääs töölaudadele lubatud. Näiteks saab rakenduse kasutamise piirata nii, et sellele pääseb ligi üksnes Tartu Ülikooli sisevõrgu kaudu või kasutades ülikooli virtuaalse privaatvõrgu (VPN) ühendust. Selline piirang on võimalik teostada AWS-i turberühmade ja virtuaalse privaatpilve (ingl *Virtual Private Cloud*, VPC) tulemüüri reeglite abil, määrates lubatud IP-aadressivahemikud. See lähenemine lisaks täiendava kaitsekihi võimalike volitamata ligipääsude vastu, ilma et oleks vaja koheselt rakendada keerukamaid autentimis- või krüpteerimismeetmeid. Samuti võib seda kasutada ajutise või täiendava kaitsemehhanismina senikaua, kuni TLS-turvalisus on rakendatud.

Kokkuvõte

Bakalaureusetöö eesmärk oli luua konteineripõhine platvorm erinevate kliimaandmete jagamiseks ja visualiseerimiseks, mis võimaldab luua konfigureeritavaid ja interaktiivseid vaateid, mis sobivad nii spetsiifilisteks rakendusteks kui ka prototüüpimiseks. Eesmärgi täitmiseks loodi Atlas, mis on mitmest komponendist koosnev süsteem, mille keskmes on isikupärastatud töölaudu käivitavad konteinerid. Süsteem sisaldab autentimismoodulit, konteinerite haldust, kasutaja sessioonihaldust ning logimis- ja monitoorimisvõimalusi. Platvorm võimaldab kasutada nii lokaalseid kui ka pilves asuvaid andmeid ning on ehitatud skaleeritavalt, kasutades Amazon Web Services teenuseid.

Viidatud kirjandus

[1] Yang S., Madsen S. M., Bednar J. A. HoloViz: Visualization and Interactive Dashboards in Python. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 2022, pp. 4846–47. ACM Digital Library, https://doi.org/10.1145/3534678.3542621.

[2] Panel. Panel v1.5.4. https://panel.holoviz.org/index.html (3.12.2024).

[3] Holoviz. HoloViz 0.17.4 Documentation. https://holoviz.org/ (3.12.2024).

[4] GeoParquet. https://geoparquet.org/ (5.12.2024).

[5] Overview. Apache Parquet. https://parquet.apache.org/docs/overview/ (5.12.2024).

[6] About GeoPandas. GeoPandas 1.0.1+0.G747d66e.Dirty Documentation. https://geopandas.org/en/stable/about.html (5.12.2024).

[7] Input/Output. Pandas 2.2.3 Documentation. https://pandas.pydata.org/docs/reference/io.html#parquet (6.12.2024).

[8] Ritter S., Ruth M. GeoTIFF Spec. http://geotiff.maptools.org/spec/geotiffhome.html (6.12.2024).

[9] Cloud Optimized GeoTIFF. https://cogeo.org/ (6.12.2024).

[10] Rasterio: Access to Geospatial Raster Data. Rasterio 1.4.3 Documentation. https://rasterio.readthedocs.io/en/stable/ (6.12.2024).

[11] Xarray. Open_rasterio. https://xarray.pydata.org/en/latest/generated/xarray.open_rasterio.html (6.12.2024).

[12] Zarr. Cloud-Optimized Geospatial Formats Guide. https://guide.cloudnativegeo.org/zarr/intro.html (13.04.2025).

[13] Introduction to the Zarr format. Copernicus Marine Help Center. https://help.marine.copernicus.eu/en/articles/10401542-introduction-to-the-zarr-format (02.05.2025).

[14] What is Amazon Cognito? Amazon Cognito. https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html (04.03.2025).

[15] Working with user attributes. Amazon Cognito. https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-attributes.html#user-pool-settings-custom-attributes (18.03.2025).

[16] ListUsers. Amazon Cognito User Pools. .https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_ListUsers.html#API_ListUsers_RequestSyntax (18.03.2025).

[17] What is Amazon Elastic Container Service? Amazon Elastic Container Service. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html (16.03.2025).

[18] Amazon ECS task definitions. Amazon Elastic Container Service. https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definitions.html?icmpid=docs_ecs_hp-task-definition (31.03.2025).

[19] AWS Glossary. https://docs.aws.amazon.com/glossary/latest/reference/glos-chap.html (06.05.2025).

Lisad

Ⅰ. Link programmi lähtekoodile

Link programmi lähtekoodile: https://gitlab.cs.ut.ee/jaannigu/ut-climate-services-portal

Ⅱ. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Jaan Nigul**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Konteineripõhise keskkonna loomine TÜ kliimateenuste portaali jaoks**, mille juhendaja on Andres Luhamaa, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi 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 ja kasutada 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.

Jaan Nigul

15.05.2025