TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

Veebirakenduse arendamine AWS-teenuste abil

Web application development using AWS

ARUKAD SÜSTEEMID JA RAKENDUSINFOTEHNOLOOGIA ÕPPEKAVA
LÕPUTÖÖ

Üliõpilane: Vjatšeslav Trelin
Üliõpilaskood: 207717EDTR
Juhendaja: Natalja Ivleva, lektor

Kohtla-Järve, 2023

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Vjatšeslav Trelin (sünnikuupäev: 10.06.2001)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Veebirakenduse arendamine AWS-teenuste abil, mille juhendaja on Natalja Ivleva,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

SISUKORD

LÜHENDITE JA TÄHISTE LOETELU

Amazon EC2 - Amazon Elastic Compute Cloud, veebiteenus, mis pakub arvutusvõimsust

Amazon ECR - Amazon Elastic Container Registry, privaatne konteinerite hoidla

Amazon ECS - Amazon Elastic Container Service, konteinerihaldusteenus

Amazon Fargate – Serverita arvutusmootor konteinerite jaoks

Amazon IAM – Amazon Identity and Access Management

Amazon S3 - Amazon Simple Storage Service

Amazon VPC - Amazon Virtual Private Cloud

AWS - Amazon Web Services, pilveteenuste platvorm

HTTP – HyperText Transfer Protocol

IAM - AWS Identity and Access Management

Konteiner - Isoleeritud kaasaskantav keskkond, mis sisaldab kõiki vajalikke faile ja teeke rakenduse käivitamiseks

ORM – Object-Relational Mapping

SSH - Secure Shell

URL - Uniform Resource Locator

Ämber – Amazon S3 objektide hoidla

SISSEJUHATUS

Selle töö kirjutamise ajal on Amazon Web Services (AWS) pilveteenuste turul liidripositsioonil, omades turust 32% [1]. AWS pakub laias valikus erinevaid teenuseid, sealhulgas arvutusvõimsust, salvestusruumi, andmebaase, analüütikat, arvutivõrke, turvalisust ja paljusid teisi. Kokku on saadaval üle 200 erineva teenuse ja tööriista [2].

Platvormi peamisteks eelisteks on madal hind, suhteline kasutuslihtsus ja lihtsustatud interaktsioon erinevate AWS tööriistade vahel. Paljud komponendid on täielikult automatiseeritud ega vaja eelnevat seadistamist. Kõik need aspektid teevad AWS-i tarkvaraarendajatele atraktiivseks valikuks.

Selle töö eesmärgiks on demonstreerida praktilise näite varal AWS-i võimalusi, mis võivad olla kasulikud tarkvaraarenduses. Töö autor valis välja hulga tööriistu, mis muudavad arendusprotsessi palju lihtsamaks ja kiiremaks, suurendades samal ajal veataluvust, turvalisust ja skaleerimist.

Lisaks moodustab suur osa tööst demo jaoks veebirakenduse loomine. Üks pool sellest on Java ja Spring Booti abil back-endi arendus. Back-end rakendus pakub kliendiga suhtlemiseks API-d, ning kasutab andmete salvestamiseks Amazon S3 ja PostgreSQL. Teine pool on front-end veebileht, mis on loodud Vue.js-iga.

Rakenduse mõlemad osad konteineriseeritakse Dockeri abil. See protsess ühendab rakenduse kõik vajalikud failid koos teekidega ning loob isoleeritud konteineri, mida saab käivitada mis tahes infrastruktuuris.

Arendustsükkel on üles ehitatud järgmiselt:

1) rakendus saab värskenduse ja uus versioon pakitakse Docker-i abil konteinerisse;
2) konteiner laaditakse üles Amazon ECR-i;
3) Amazon ECR-is asuv konteinerrakendus juurutatakse Amazon EC2 peal Docker-i abil.

Võtmesõnad: AWS, Cloud, Docker, Java, API, bakalaureusetöö.

1. AWS-TEENUSTE ÜLEVAADE

Tänapäeva maailmas nõuab veebirakenduste arendamine suurt paindlikkust, skaleerimist ja töökindlust. Pilvandmetöötlusest on saanud selle protsessi lahutamatu osa, mis annab arendajatele juurdepääsu võimsatele tööriistadele ja teenustele.

Amazon Web Services, üks juhtivaid pilvandmetöötluse platvorme, pakub arendajatele laia valikut teenuseid ja ressursse, võimaldades neil luua ja juurutada veebirakendusi minimaalse pingutusega.

Arvestades alternatiivide suurt hulka, on oluline valida õiged teenused, lähtudes projekti konkreetsetest vajadustest ja nõuetest.

Selles osas vaadeldakse mitmeid populaarseid AWS-teenuseid, mis ei vaja keerukamat konfigureerimist ja on seetõttu kasulikud arendajatele, kes ei soovi tegeleda liigse infrastruktuuri haldamisega.

1.1 Käivitamine ja juurutamine

1.1.1 Amazon Elastic Compute Cloud

Amazon EC2 pakub pilvepõhiseid virtuaalmasinaid, mida arendajad saavad konfigureerida ja hallata. See võimaldab täielikku kontrolli virtuaalmasina infrastruktuuri üle, muutes selle populaarseks valikuks.

- Täielik kontroll: EC2 pakub suurel määral paindlikkust ja täielikku juurdepääsu virtuaalsetele masinatele. Arendajad saavad kohandada operatsioonisüsteeme, installida vajalikku tarkvara ja konfigureerida võrgusätteid. Selle kontrollitasemega kaasneb aga ka suurem vastutus virtuaalmasinate haldamise eest.
- Skaleeritavus: EC2 võimaldab ressursse paindlikult skaleerida, sealhulgas eksemplaritüüpide muutmist, eksemplaride arvu suurendamist või vähendamist ning automaatse skaleerimise konfigureerimist. See on eriti kasulik muutuva koormusega veebirakenduste puhul.
- ECS-i integratsioon: EC2 integreerub otse Amazon ECS ja EKS-iga ning võimaldab käivitada ja hallata konteinereid ECS või EKS klastri kaudu.

1.1.2 Amazon Elastic Beanstalk

Amazon Elastic Beanstalk on AWS teenus, mis lihtsustab rakenduste juurutamist ja haldamist pilves. Elastic Beanstalk pakub arendajatele hõlpsasti kasutatavat platvormi rakenduste juurutamiseks ja käivitamiseks, ilma infrastruktuuri keerukusega tegelemata. Elastic Beanstalk toetab mitut programmeerimiskeelt, sealhulgas Java,

.NET, PHP, Node.js, Python, Ruby, Go ja Dockeri konteinereid, muutes selle mitmekülgseks mitmesuguste rakenduste jaoks. [3]

Teenuse eelised:

- Kasutuslihtsus: Elastic Beanstalk eemaldab suure osa infrastruktuuri haldusest, muutes arendajatel hõlpsaks rakenduste juurutamise, ilma et nad peaksid keerukatesse seadistustesse süvenema.
- Automaatne skaleerimine: Elastic Beanstalk saab teie rakendust vastavalt nõudlusele automaatselt skaleerida, tagades, et see saab hakkama erineva töökoormusega ilma käsitsi sekkumiseta.
- Hallatud keskkond: AWS hoolitseb aluseks oleva infrastruktuuri, sealhulgas serverite varustamise, koormuse tasakaalustamise ja võimsuse planeerimise eest, võimaldades arendajatel keskenduda oma rakenduste loomisele ja täiustamisele.
- Integreeritud AWS-teenustega: Elastic Beanstalk integreerub sujuvalt teiste AWS-teenustega, nagu Amazon RDS, Amazon S3 ja palju muud, pakkudes terviklikku platvormi skaleeritavate rakenduste loomiseks.

Teenuse puudused:

- Piiratud kohandamine: Kuigi Elastic Beanstalk lihtsustab juurutamist, ei pruugi see sobida väga kohandatud või keerukate infrastruktuuride jaoks. Spetsiifiliste infrastruktuurinõuetega arendajatel võib see abstraktsioon olla piiratud.
- Sõltuvus AWS-ist: Elastic Beanstalk seob kasutajad AWS-i ökosüsteemiga. Kuigi see on kasulik kasutajatele, kes on juba AWS-i investeerinud, võib see olla puuduseks neile, kes otsivad agnostilisemat lahendust.

1.1.3 Amazon App Runner

Amazon App Runner on täielikult hallatav AWS-teenus, mis on loodud rakenduste lihtsustatud ja kiireks juurutamiseks. See automatiseerib koostamise, juurutamise ja skaleerimise protsessid, võimaldades arendajatel keskenduda programmeerimisele, saades samas kasu automaatsest infrastruktuurihaldusest.

Teenuse plussid:

- Täielikult hallatud: App Runner tegeleb juurutamise, skaleerimise ja infrastruktuuri haldamisega.
- Lihtne juurutamine: lihtsustab juurutamist lähtekoodist või konteineri kujutistest.

- Automaatne skaleerimine: skaleerib dünaamiliselt koormuse põhjal, et tulla toime erineva töökoormusega.
- Integratsioon: sujuv integreerimine teiste AWS-teenustega täiustatud funktsionaalsuse tagamiseks.

Teenuse puudused:

- Piiratud kohandamine: ei pruugi sobida väga kohandatud või keerukate infrastruktuuride jaoks.
- Piiratud konteinerite orkestreerimine: väga põhilised orkestreerimisfunktsioonid.

1.2 Konteinerite orkestreerimine ja haldamine

1.2.1 Amazon Elastic Container Service

Amazon ECS on hallatav konteinerite orkestreerimisteenus, mis on loodud Dockeri-põhiste konteinerite juurutamiseks ja haldamiseks. ECS loob automaatselt vajalikud EC2 või Fargate eksemplarid ja teostab konfiguratsiooni. [4]

Amazon ECS on atraktiivne järgmistel põhjustel:

- Skaleeritavus: Amazon ECS muudab konteinerite skaleerimise lihtsaks.
- Integreerimine teiste teenustega: ECS integreerub sujuvalt teiste AWS-teenustega, võimaldades teil arendada tugevaid veebirakendusi.

1.2.2 Amazon Elastic Kubernetes Service

Amazon EKS pakub hallatavat Kubernetese klastrit ja suuremaid konteinerite orkestreerimisvõimalusi kui ECS. Kubernetes on ka praegune tarkvaraarenduse de facto standard. EKS nõuab aga ulatuslikumat seadistamist ja haldamist ning sellel pole tasuta kasutamisvõimalust. [5]

1.2.3 Amazon Fargate

Amazon Fargate on kõrgetasemeline konteineri käivitamise teenus, mis pakub abstraktsiooni virtuaalmasinate haldamiseks. Amazon ECS-i või EKS-i kasutajad saavad Fargate'iga konteinereid käivitada ja AWS hoolitseb automaatselt arvutusressursside eest. See vabastab arendajad virtuaalmasina infrastruktuuri haldamisest, lihtsustades konteinerite arendamise ja juurutamise protsessi. [6]

- Kasutuslihtsus: Amazon Fargate on kõrgelt automatiseeritud ja muudab konteineritega töötamise lihtsaks. Arendajad saavad serverite haldamise asemel keskenduda rakendusele ja konteineritele.

- Ressursihaldus: Fargate haldab automaatselt ressursse, sealhulgas protsessoreid ja mälu, konteinerinõuete alusel. See võimaldab optimeerida ressursside kasutamist ja vähendada kulusid.

EC2 pakub suuremat paindlikkust ja kontrolli, kuid nõuab rohkem haldust, Fargate pakub aga kasutusmugavust ja automatiseerimist, kuid on mõnes kohandamise aspektis piiratud.

1.3 Salvestustehnoloogiad

1.3.1 Amazon Relational Database Service

Amazon RDS on täielikult hallatav relatsiooniandmebaasi teenus, mida pakub Amazon Web Services (AWS). See võimaldab kasutajatel pilves relatsiooniandmebaase seadistada, kasutada ja skaleerida, ilma käsitsi sekkumiseta. RDS toetab erinevaid andmebaasimootoreid, nagu MySQL, PostgreSQL, Oracle, SQL Server ja MariaDB, pakkudes usaldusväärset ja skaleeritavat lahendust erinevate rakenduste vajaduste jaoks.

Amazon RDS tegeleb automaatselt rutiinsete andmebaasiülesannetega, tarkvara parandamise ja riistvaraga, vähendades sellega seotud halduskulusid. Samuti pakub see teenus igapäevaseid varukoopiaid õigeaegse taastamise võimalustega, tagades andmete vastupidavuse ja hõlbustades lihtsat taastamist.

Andmebaasi eksemplare on lihtne skaleerida vertikaalselt (riistvara uuendamine) või horisontaalselt (andmete kopeerimine mitme eksemplari vahel), et rahuldada kasvavat töökoormust. Multi-AZ (kättesaadavuse tsooni) juurutused pakuvad andmebaasi eksemplari tõrke korral kõrget kättesaadavust ja automaatset tõrke lahendamist. Andmekaitse eesmärgideks on ka palju turvafunktsioone, nagu krüpteerimine, IAM-i (Identity and Access Management) integreerimine ja võrgu isoleerimine.

Sellel on ka mitmeid varjukülgi: kuigi hallatav teenus vähendab töö keerukust, on kasutajatel vähem kontrolli all oleva infrastruktuuri üle võrreldes isehostitavate andmebaasilahendustega. Samuti võivad kulud olenevalt valitud andmebaasimootorist ja eksemplari suurusest kiiresti suureneda, eriti suuremahuliste või ressursimahukate rakenduste puhul.

1.3.2 Amazon Simple Storage Service

Amazon S3 on pilvepõhine andmehoidla, mis on loodud suurte andmemahtude, sealhulgas veebirakenduste jaoks vajalike staatiliste ressursside salvestamiseks ja haldamiseks. Amazon S3 on valitud järgmistel põhjustel:

- Usaldusväärne andmesalvestus: S3 tagab andmete salvestamisel kõrge

kättesaadavuse ja töökindluse, mistõttu on see mõistlik valik staatiliste rakendusressursside jaoks.

- Skaleeritavus: S3 võimaldab paindlikult skaleerida andmesalvestust vastavalt vajadusele, ilma infrastruktuuri haldamiseta.
- Paindlik juurdepääsukontroll: S3 pakub mitmesuguseid andmetele juurdepääsu juhtimise võimalusi, sealhulgas avalikku juurdepääsu ja võtmepõhist juurdepääsu.

Salvestusklassi valimine:

Amazon S3-l on palju salvestusklasse, mis on loodud erinevate salvestusstrateegiate jaoks (vt Joonis 1.1).

Performance across the S3 storage classes

	S3 Standard	S3 Intelligent-Tiering*	S3 Express One Zone**	S3 Standard-IA	S3 One Zone-IA**	S3 Glacier Instant Retrieval	S3 Glacier Flexible Retrieval***	S3 Glacier Deep Archive***
Use cases	General purpose storage for frequently accessed data	Automatic cost savings for data with unknown or changing access patterns	High performance storage for your most frequently accessed data	Infrequently accessed data that needs millisecond access	Re-creatable infrequently accessed data	Long-lived data that is accessed a few times per year with instant retrievals	Backup and archive data that is rarely accessed and low cost	Archive data that is very rarely accessed and very low cost
First byte latency	milliseconds	milliseconds	single-digit milliseconds	milliseconds	milliseconds	milliseconds	minutes or hours	hours
Durability	Amazon S3 provides the most durable storage in the cloud. Based on its unique architecture, S3 is designed to exceed **99.999999999% (11 nines)** data durability. Additionally, S3 stores data redundantly across a minimum of 3 Availability Zones by default, providing built-in resilience against widespread disaster. Customers can store data in a single AZ to minimize storage cost or latency, in multiple AZs for resilience against the permanent loss of an entire data center, or in multiple AWS Regions to meet geographic resilience requirements.							
Designed for availability	99.99%	99.9%	99.95%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	1	≥3	1	≥3	≥3	≥3
Minimum storage duration charge	N/A	N/A	1 hour	30 days	30 days	90 days	90 days	180 days
Retrieval charge	N/A	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
Lifecycle transitions	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes

Joonis 1.1 Amazon S3 salvestusklassid [7]

Selle töö vajadusteks valiti salvestusklass Standard. See variant pakub andmete vastuvõtmisel madalat latentsust, mis aitab veebilehti kiiresti laadida ja parandab kasutajakogemust. Amazon S3 Standard pakub ka piisavalt töökindlust rakenduse kasutamiseks.

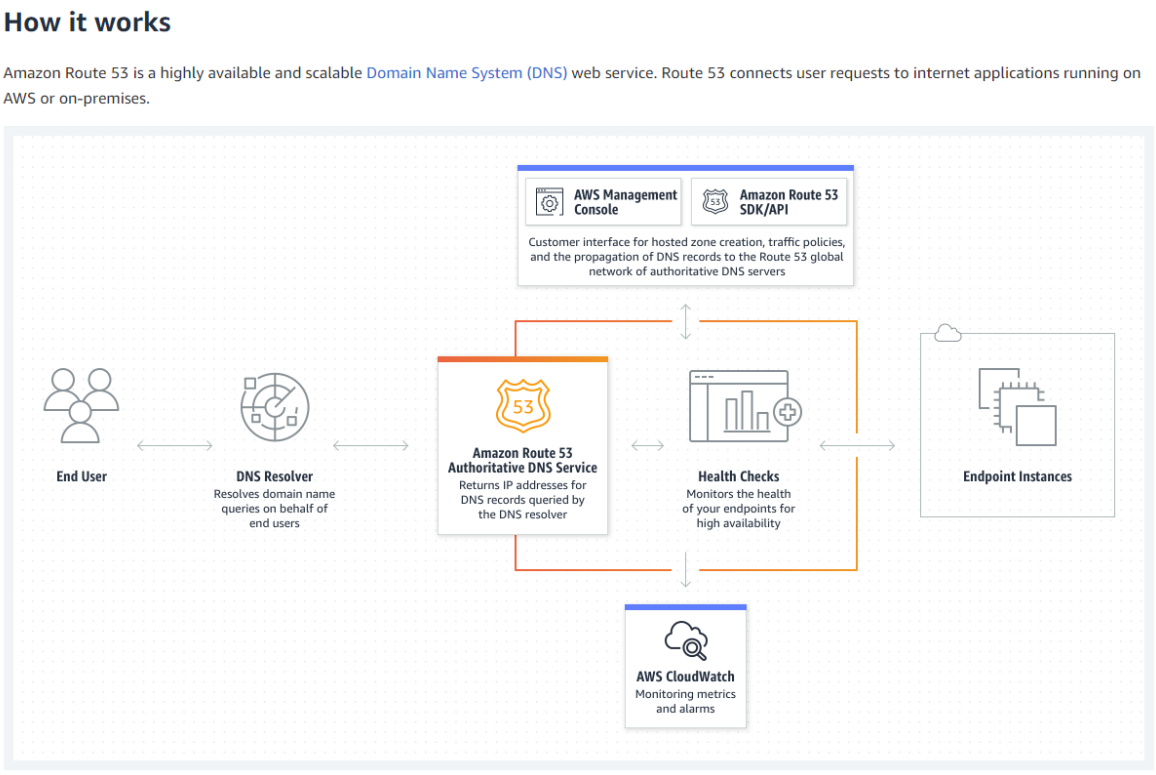
2. AWS-TEENUSTE VALIMINE

Amazon Web Service-i poolt pakutavate erinevate teenuste rohkus võib raskendada valiku tegemist isegi IT-professionaali jaoks, kellel on hea arusaam nõutavast arvutusvõimsusest. Selles peatükis pakub autor mitmeid erinevaid AWS-i profiile erinevate kasutusmastaapide jaoks.

Kõik need profiilid pakuvad soovitatavate konfiguratsioonidega teenuste loendit, mida on võimalik kasutada selle töö järgmises peatükis oleva näitega sarnase projekti arendamiseks.

Need profiilid on lähtepunktiks arendamiseks ja neid soovitakse hiljem muuta vastavalt rakenduse konkreetsetele vajadustele. Samuti on oluline konfigureerida sobivad turberühmad, virtuaalsed võrgud, IAM rollid ja rakendada dokumentatsioonis soovitatud kasutusviise iga pakutud profiilides kasutatava AWS-teenuse jaoks.

Kõiki neid profiile saab täiendada Amazon Route 53-ga, kui on vaja kasutada domeeninimede süsteemi (DNS), või Amazon Elastic IP teenusega, kui on vaja saada staatilisi IP-aadresse.



Joonis 2.1 Amazon Route 53 seadistamise näidis [8]

2.1 Isiklikuks kasutamiseks – Tasuta profiil

Esimine profiil on kõige paindlikum variant ja ka see, mida kasutatakse järgmistel demonstratsioonidel. See profiil pakub suurimat vabadust, kuid sellel on piirang, mis

nõuab virtuaalmasina käsitsi seadistamist.

Amazon Free Tier pakub paljude teenuste tasuta kasutamist 12 kuud, mis on hea testimiseks, õppimiseks või isiklike tööriistade hostimiseks.

Selle valiku korral hostitakse veebirakendust täielikult ühel EC2 eksemplaril. Turvagruppide seadistamine võimaldab piirata juurdepääsu virtuaalsele masinale ainult teatud portides, mis lisab turvakihi. Tänu SSH kaudu virtuaalmasinale ligipääsule on võimalik kõik rakenduse käivitamiseks vajalikud tööriistad käsitsi installida. Näites kasutatakse piltide salvestamiseks ka Amazon S3.

Kasutatav EC2 eksemplari tüüp on t2.micro või t3.micro, kuna need kuuluvad Amazon EC2 tasuta kasutustingimuste alla [9].

2.2 Isiklikuks kasutamiseks – Tasuline profiil

See profiil sobib väikese eelarvega kasutamiseks. Tasuliste teenuste, nagu Amazon App Runner ja Amazon RDS, kasutamine annab võimalust teie rakenduse juurutamise ja hooldamise protsessi lihtsustada. See valik suurendab ka usaldusväärsust, pakkudes võimalusi andmebaasi varundamiseks, samuti eraldades andmebaasi rakendusest, võimaldades rakendust tulevikus skaleerida.

See profiil soovitab kasutada piltide salvestamiseks Amazon S3, rakenduse käivitamiseks Amazon App Runnerit ja andmebaasina Amazon RDS-i. Amazon Elastic Beanstalk on ka hea alternatiiv App Runnerile, aga vajab rohkem aega sügavamale kohandamisele.

Kuna Amazon Free Tier pakub iga kuu 750 tundi tasuta Amazon RDS-i kasutamist [9], pakub praegune profiil järgmist konfiguratsiooni:

- Single-AZ
- Instance Type: db.t2.micro / db.t3.micro / db.t4g.micro

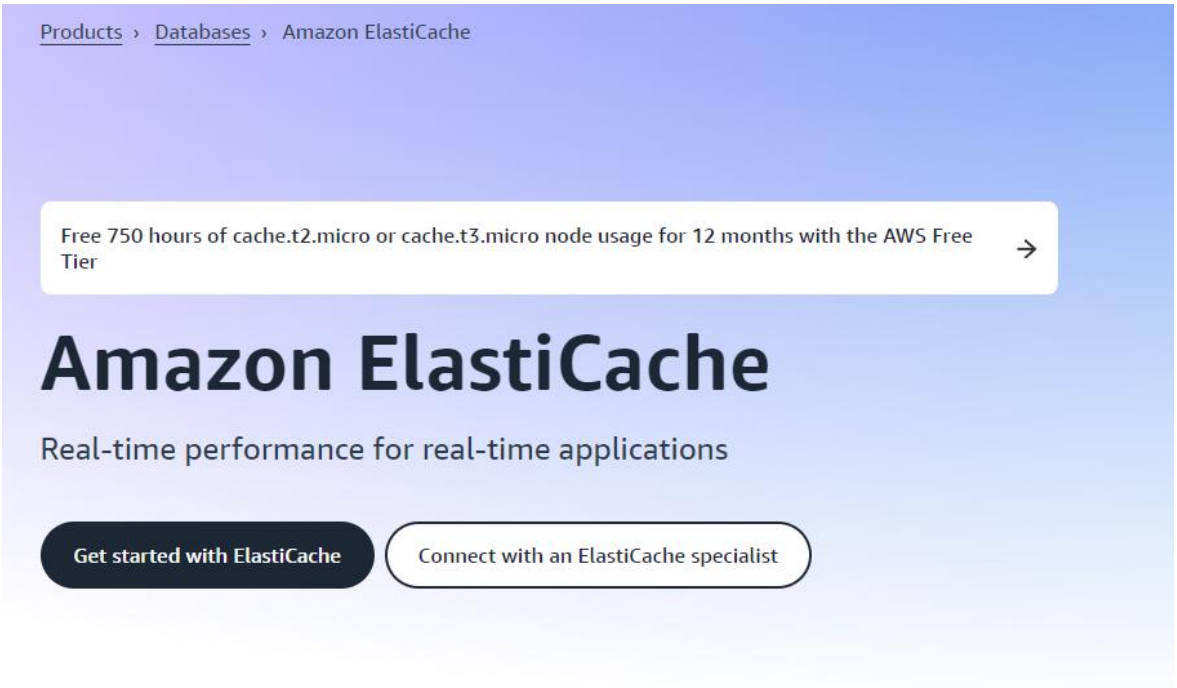
2.3 Väikeettevõte – Tasuline profiil

Selles profiilis soovitab autor pöörata tähelepanu rakenduse jõudlusele. Korrates suuresti eelmist profiili, soovitakse siin valida Amazon Elastic Beanstalk selle paindlikumate konfigureerimis- ja skaleerimisvõimaluste tõttu.

Piltide salvestamiseks kasutatakse Amazon S3 ja relatsiooniandmebaasina on Amazon RDS. Andmemahu kasvades on mõistlik valida suurem kettaruumi maht ning valida ka kiirem mälutüüp, näiteks gp2 asendada gp3-ga.

Siinkohal oleks hea lisada vahemälu tugi Amazon ElastiChache või mõne alternatiivi abil. Amazon ElastiCache for Redis võimaldab teil ilma suuremate muudatusteta

kasutada rakendusi, mis juba kasutavad Redis-i, lisades samas funktsioonid vahemälusõlmede automaatseks tuvastamiseks ja parandamiseks.



Joonis 2.2 Amazon ElastiCache teenuse kirjeldus [10]

2.4 Suurettevõte – Tasuline profiil

Ettevõtte kasvades kasvavad ka nõuded seda toetavatele infosüsteemidele. Praeguses etapis ei ole konkreetse ettevõtte vajadusi teadmata enam võimalik 100% tõenäosusega ühtegi soovitust anda. See profiil on eelkõige edasise kasvu lähtepunkt.

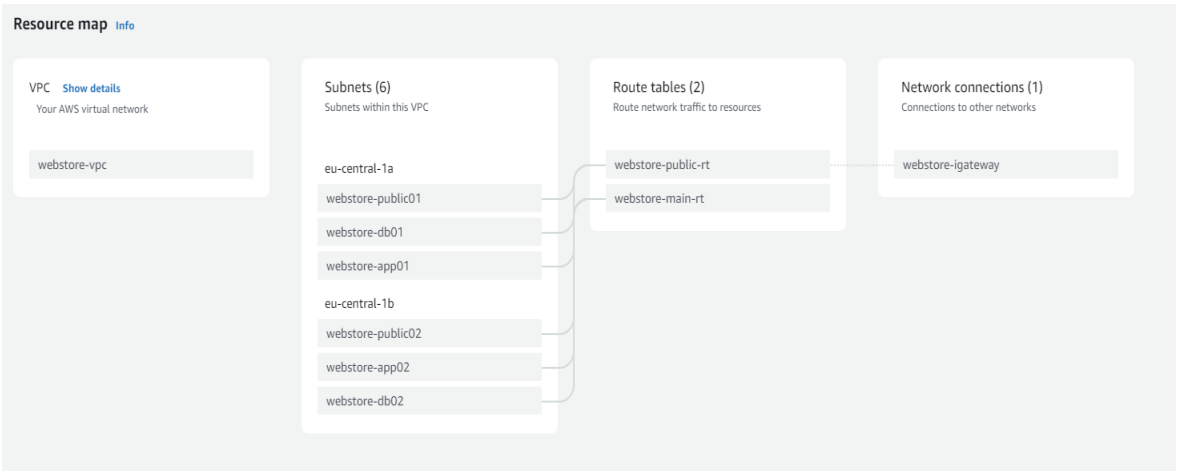
Suurettevõte profiil soovitab kasutada konteinerite orkestreerimissüsteeme, nagu Amazon Elastic Container Service või Amazon Elastic Kubernetes Service, kui ettevõttes on piisav kogemust Kubernets-iga töötamiseks. Orkestreerimissüsteemide seadistamisel saab valida Amazon EC2 ja Amazon Fargate'i vahel, kuid automaatse seadistamise tõttu soovitab autor valida Amazon Fargate'i.

Selles etapis muutuvad kohustuslikuks ka koormuse tasakaalustamine ja vahemällu salvestamine, sest orkestratsioonisüsteemid loovad palju rakenduse eksemplare ning rakenduse kui terviku jõudlus muutub väga oluliseks.

Autori arvates on Amazon S3 koos Amazon CloudFrontiga üks parimaid valikuid staatiliste piltide kasutamiseks. Tänu vahemällu salvestamisele pakub CloudFront kiiret juurdepääsu S3-sse salvestatud ressurssidele mis tahes piirkonnast, olenemata algsest asukohast.

Amazon RDS-i pakutakse endiselt andmete salvestamiseks, kuid andmebaasi konfiguratsioon muutub kriitiliseks. Soovitatav on kulutada rohkem aega võrgu

seadistamisele ja turvalisuse tagamisele (vt Joonis 2.3).



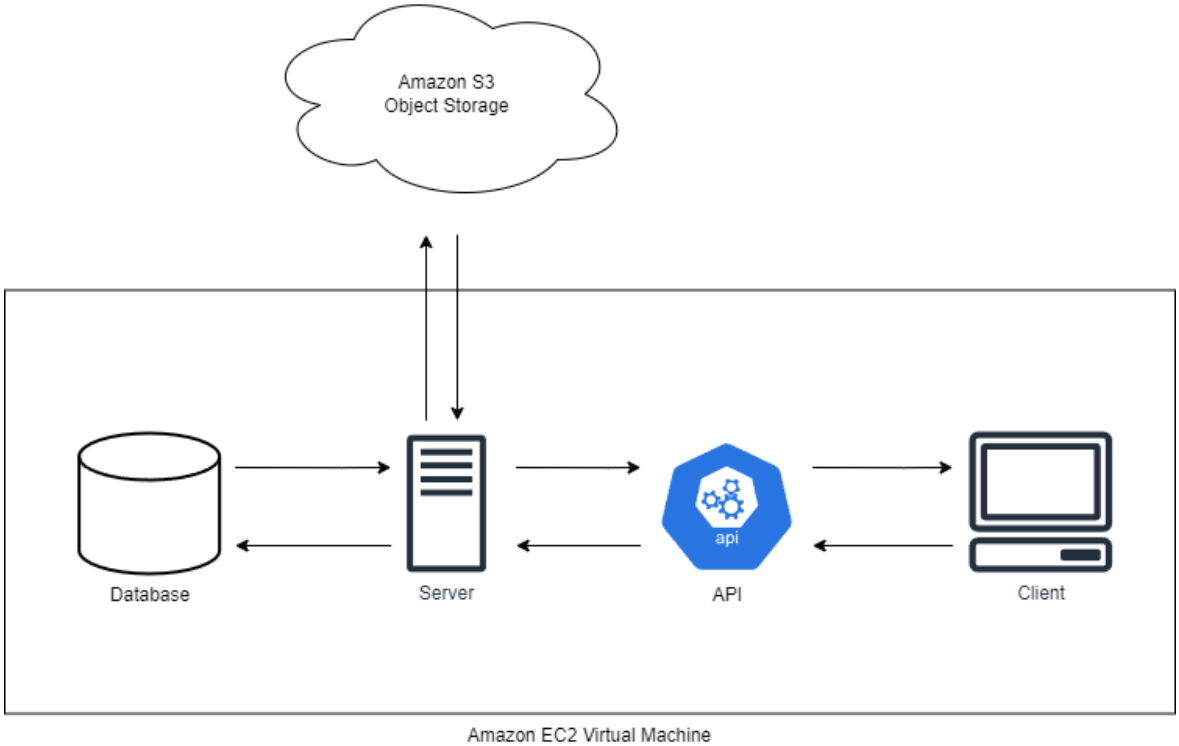
Joonis 2.3 Võrgu konfiguratsiooni näidis

3. PROJEKTEERIMINE JA ARENDUS

AWS-i teenuste kasutamise demonstreerimiseks lõi autor veebipoe väikese prototüübi. Kuna rakenduse eesmärk on demonstreerida AWS-i võimalusi, otsustas autor seda mitte siduda ühegi teise teenusega peale Amazon S3 ja Amazon EC2 ning samuti mitte laiendada funktsionaalsust kaugemale CRUD põhifunktsioonide ja autoriseerimisvõimaluste tasemest.

Prototüüp on hajutatud rakendus, mis on üles ehitatud klient-serveri mudelile ja koosneb kahest põhiosast:

- Server: back-end rakendus, mis pakub kliendiga suhtlemiseks API-d. Klient saab kasutada API päringuid andmebaasi salvestatud ressurssidele juurdepääsu saamiseks. Serveril on juurdepääs ka Amazon S3-le.
- Klient: front-end rakendus, mis pakub graafilist liidest lõppkasutajaga suhtlemiseks. Selle rakenduse funktsioone kasutades saadab kasutaja API päringuid serveri hallatavatele ressurssidele juurdepääsu saamiseks.



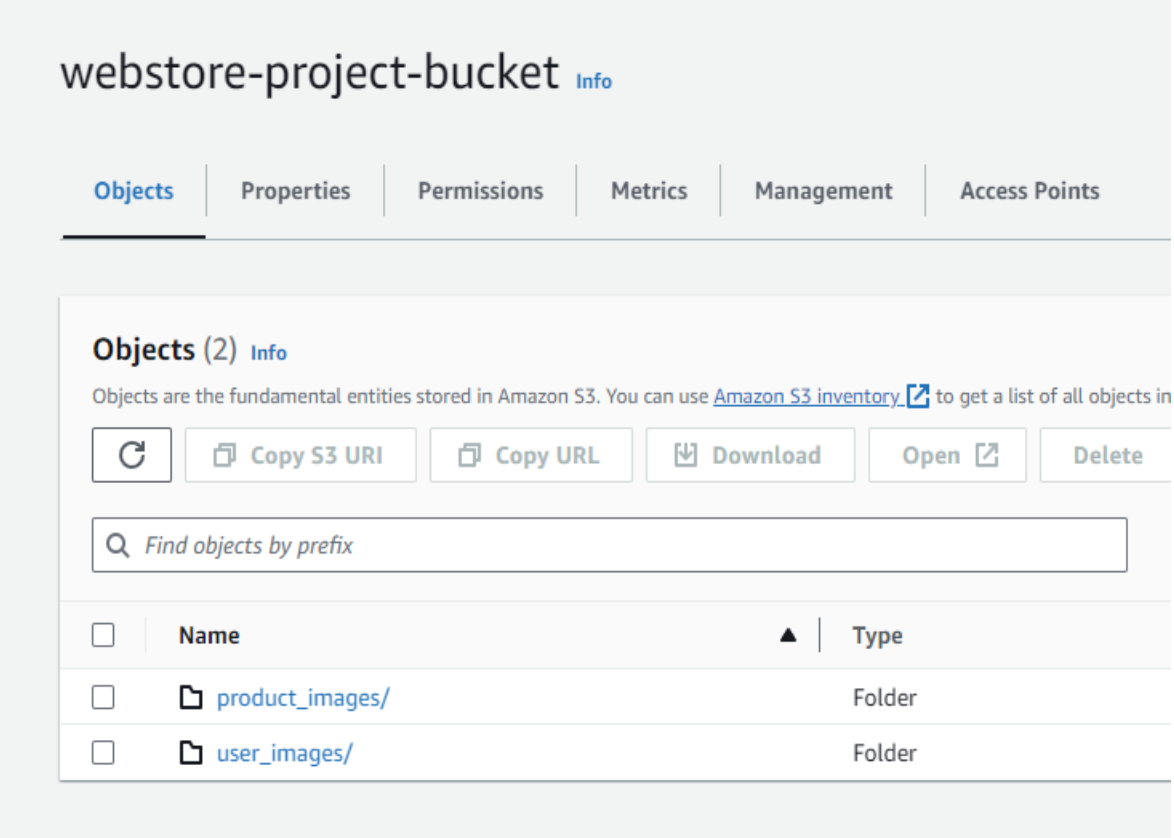
Joonis 3.1 Rakenduse diagramm

3.1 Back-end rakenduse arendamine

Back-end rakendus on kirjutatud Java programmeerimiskeeles, kasutades Spring Frameworki. Rakenduse andmed salvestatakse PostgreSQL relatsiooniandmebaasi abil. Selle valiku põhjuseks oli autori kogemus nende tehnoloogiatega töötamisel.

Andmebaasi tabelite struktuur luuakse dünaamiliselt ja seda saab muuta, kui olemitele lisatakse uusi atribuute. ORM-tehnoloogiat kasutatakse andmebaasis salvestatud olemitega töötamiseks.

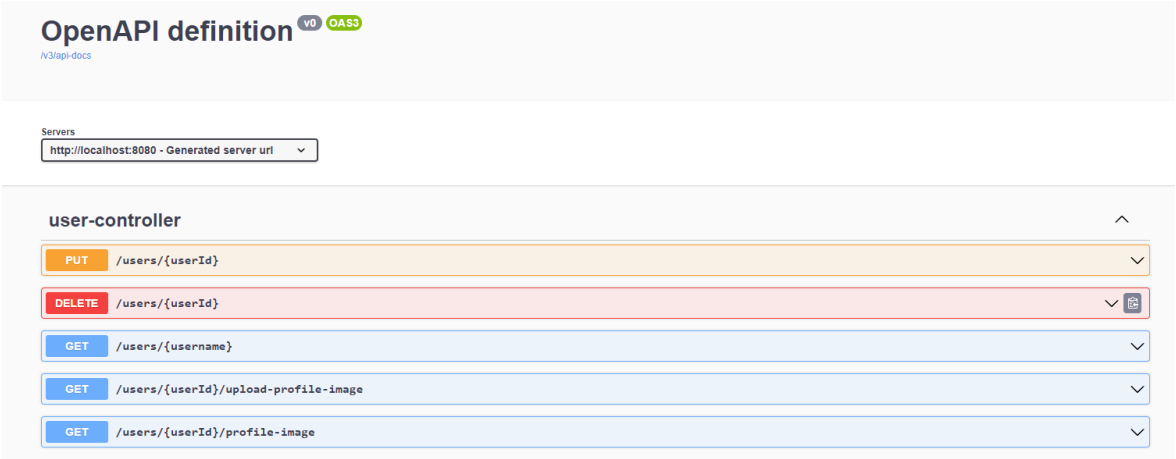
Kasutajate ja toodete pildid salvestatakse Amazon S3 pilvesalvestusse. Selleks oli eelnevalt loodud privaatne Amazon S3 ämber koos vajaliku failistruktuuri ja seadistustega. (vt Joonis 3.2).



Joonis 3.2 Amazon S3 ämbri vaade

Rakendus pakub kliendiga suhtlemiseks API-d. Päringute autoriseerimiseks kasutatakse seansihaldust ja kasutajarolle. Seega nõuavad mõned olemasolevad API lõpp-punktid kasutajalt autentimist, teised vajavad lisaks administraatori rolli.

Saadaolevate API lõpp-punktide loend ja vastavad HTTP-meetodid on volitatud kasutajatele saadaval Swaggeri kasutajaliidese abil loodud rakenduse juurtee lehel (vt Joonis 3.3).



Joonis 3.3 Swagger UI API lõpp-punktid

3.1.1 Andmebaasi struktuur ja kirjeldus

Selles konfiguratsioonis asub andmebaas back-end serveriga samas masinas. See pole arhitektuurilisest vaatenurgast ideaalne, kuna see ei võimalda rakendusel skaleerida. Sellest lähenemisviisist võib aga piisata prototüüpide, väikeste rakenduste ja isiklike tööriistade puhul, nagu tasuta profiil soovitab.

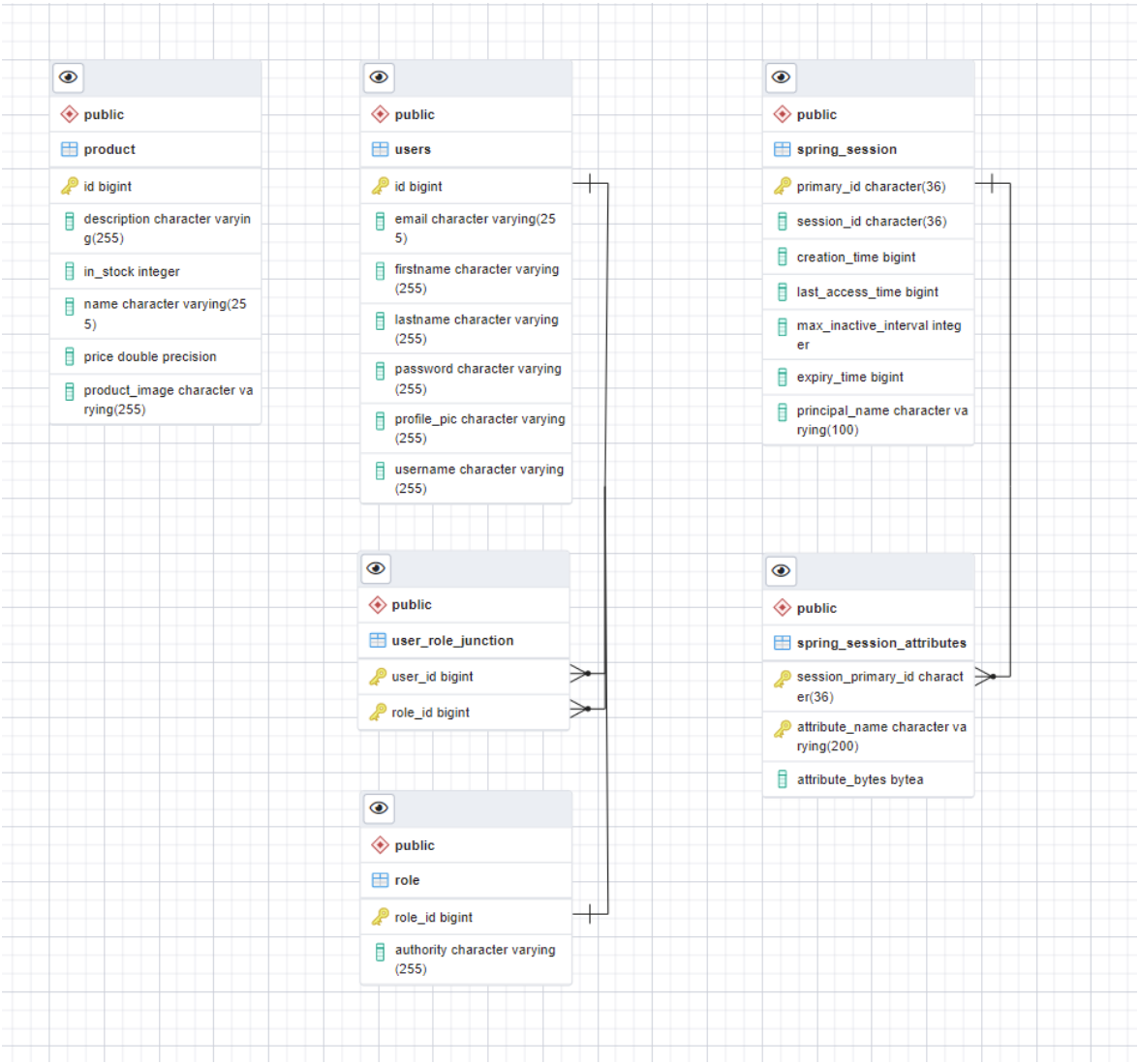
Nagu eelmises osas märgitud, kasutab taustarakendus ORM-tehnoloogiat üksuste salvestamiseks andmebaasi.

Selliseid olemeid on kokku 3:

- Product, milllele vastab taabel «product»;
- User, milllele vastab taabel «users»;
- Role, milllele vastab taabel «role»;

Samuti on kasutajarollide salvestamiseks loodud tabel “user_role_junction”.

Tabelid «spring_session» ja «spring_session_attributes» on loodud Spring Sessioni kasutamise tõttu. Nad salvestavad andmeid praeguste seansside kohta, nii et kasutajad saaksid jätkata rakenduse kasutamist ka pärast serveri taaskäivitamist. Pärast taaskäivitamist hangib server nendest tabelitest andmed, võimaldades kasutajatel tööd jätkata.



Joonis 3.4 Andmebaasi ERD diagramm

3.1.2 Ühendamine andmebaasiga

Andmejuurdepääsukiht töötab Spring Data JPA abil. Selles etapis kasutatakse mitut tarkvarakujundusmustrit: Service kui vahekiht, milles toimub andmetöötlus, Repository andmetele otseseks juurdepääsuks ja Data Access Object andmete kapseldamiseks ja kasutajale edastamiseks. Kui andmetele juurdepääsu päring saabub, kasutab API kontroller vastavat Service-it, mis täidab vajaliku äriloogika. Teenus kasutab seejärel andmetele otse juurdepääsuks Repository. Data Access Object-i kasutamine on praegusel juhul tingitud turvaprobleemidest – see mall võimaldab edastada kasutajale vaid vajalikud objektiväljad.

Praeguses konfiguratsioonis hostitakse andmebaasi programmiga samas masinas, mis võimaldab ühendust localhost-i kaudu. Arengu kiirendamiseks valis autor lihtsa kasutajanime ja parooli ning režiimi ddl-auto=update, mis võimaldab muuta andmebaasi struktuuri. Kõik need seadistused tuleks muuta turvalisemaks, kui see

rakendus on kasutusel reaalsete ülesannete jaoks.

```
# Datasource
spring.datasource.name=webstore
spring.datasource.url=jdbc:postgresql://localhost:5432/webstore
spring.datasource.username=webstore
spring.datasource.password=webstore

# Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
```

Joonis 3.5 Spring Data JPA ja Hibernate konfiguratsioon application.properties failis

3.1.3 AWS-i SDK

AWS SDK lihtsustab AWS-i teenuste kasutamist, pakkudes Java arendajatele järjepidevat ja tuttavat teeke. See pakub tuge API elutsükliga arvestamiseks, nagu mandaatide haldamine, korduskatsetused, andmete jagamine ja serialiseerimine. AWS SDK for Java toetab ka kõrgema taseme abstraktsioone lihtsustatud arenduse jaoks. [11]

Selle prototüübi jaoks kasutab autor AWS SDK for Java 1.11.163, et luua ühendus Amazon S3-ga. AWS SDK kasutamiseks on vaja lisada Amazon SDK Maven-i sõltuvusena (dependency)(vt Joonis 3.6).

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
    <version>1.11.163</version>
</dependency>
```

Joonis 3.6 AWS S3 SDK for Java 1.11.163 paigaldamine kasutades Maven-i

Amazon S3-ga töötamiseks AWS SDK abil loodi S3Config konfiguratsiooniklass ja S3Service teenuseklass. See klass kasutab juurdepääsuvõtit, IAM-i kaudu loodud salajast võtit, samuti Amazon S3 ämbri piirkonda ja nime. Need sätted tuleb määrata failis application.properties. Tulevikus toimub kogu suhtlus Amazon S3-ga S3Service-i kaudu.

```java
@Configuration
public class S3Config {
    @Value("${aws.accessKey}")
    private String accessKey;

    @Value ("${aws.secretAccessKey}")
    private String secretKey;

    @Value("${aws.region}")
    private String region;

    @Bean
    public AmazonS3 s3Client(){
        final AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
        return AmazonS3ClientBuilder
                .standard()
                .withCredentials(new AWSStaticCredentialsProvider(credentials))
                .withRegion(region)
                .build();
    }
}
```

Joonis 3.7 Amazon S3-ga ühendamine kasutades AWS-i SDK

Seda klienti saab hiljem kasutada failide laadimiseks määratud Amazon S3 ämbrisse. Autori kasutatud lahenduses tagastab server kliendi nõudmisel eelallkirjastatud URL-i, millega võib saada faili lingi HTTP GET meetodiga või laadida üles uue faili, kasutades HTTP PUT abil saanud linki.

Need lingid on turvalised, kuna need ei võimalda kliendil juurde pääseda muudele failidele peale nende, mida server lubab. Need sätted võimaldavad juurdepääsu selle lingi kaudu ainult 30 minutiks, mis vähendab URL-i varguse võimalikku rünnakuakent.

Amazon S3 ämbri seadmised võimaldavad määrata üleslaaditava faili maksimaalse suuruse, et vältida olukorda, kus ründaja laadib lingi kaudu üles liiga suure faili, mis võib kaasa tuua teenuse kasutamise eest arve suurenemist.

```java
@Service
public class S3Service {
    private final AmazonS3 s3Client;
    @Value("${aws.s3.bucket}")
    private String bucketName;

    public S3Service(AmazonS3 s3Client) {
        this.s3Client = s3Client;
    }

    public String generateUrl(String filename, HttpMethod http){
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(new Date());
        calendar.add(Calendar.MINUTE,  amount: 30);
        URL url = s3Client.generatePresignedUrl(bucketName, filename, calendar.getTime(), http);
        return url.toString();
    }
}
```

Joonis 3.8 Eelallkirjastatud URL-i loomine

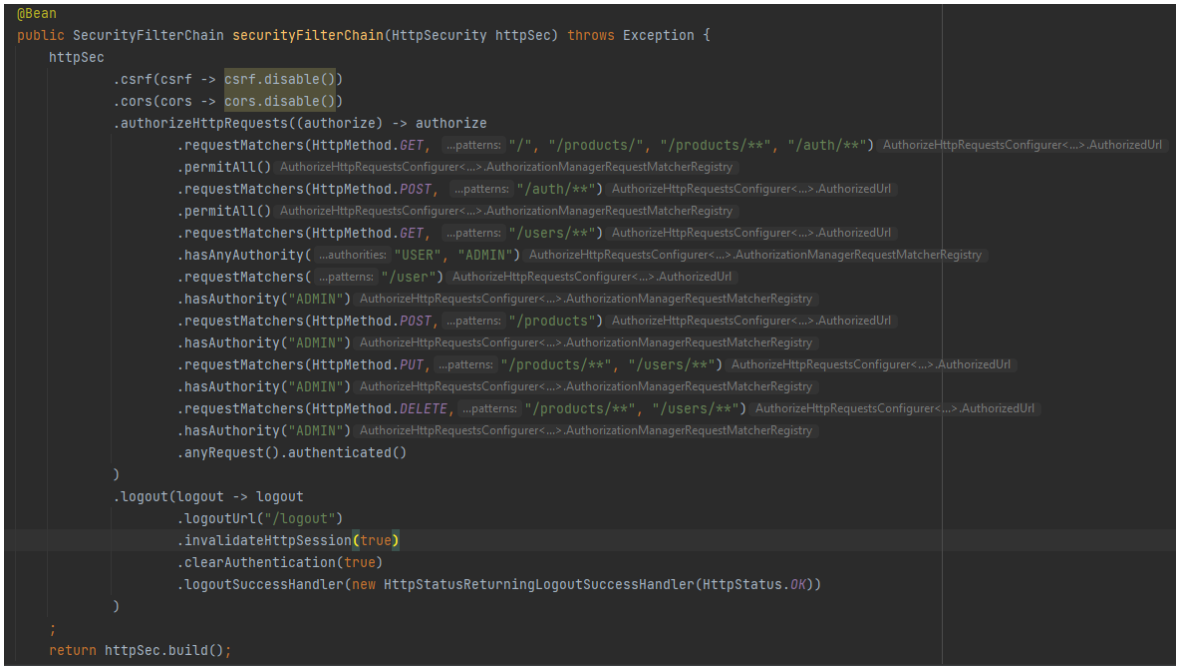
Joonis 3.9 API vastab eelallkirjastatud URL-iga

3.1.4 Autoriseerimine

API päringu esitamisel peab päring esmalt läbima Spring Security filtri. See filter piirab juurdepääsu kõigidele API lõpp-punktile (vt Joonis 3.10). Hetkel olemas, registreeritud kasutajal võib olla maksimaalselt 2 rolli – „USER", „ADMIN". „USER" rolliga kasutajatel ei ole õigust teist kasutajaid ega tooteid muuta. Kasutajatel, kelle roll on „ADMIN", on maksimaalne juurdepääsutase.

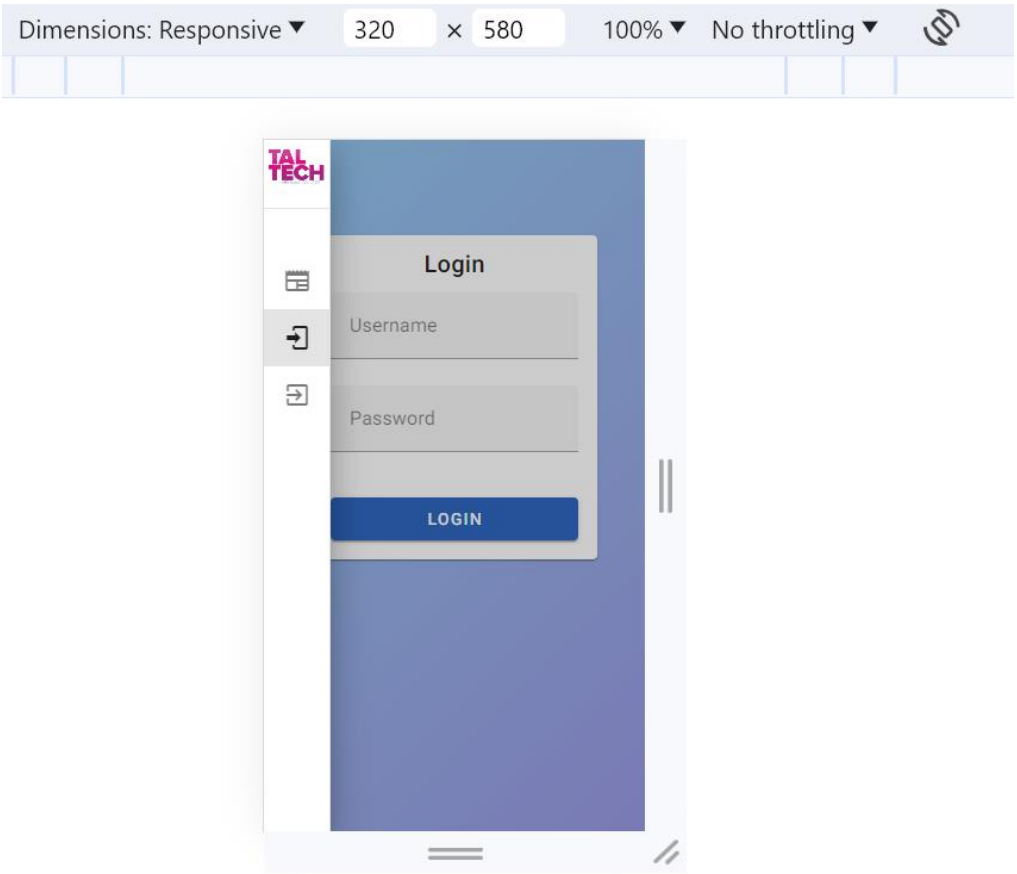
Autentimata kasutajatel on ainult mõned lubatud toimingud: avalehe vaatamine, toodete vaatamine, sisselogimine ja registreerimine.

Kui päring seda filtrit ei läbi, saab kasutaja serverilt vastusena HTTP koodi 403 – Forbidden.



Joonis 3.10 Rakenduse turvalisuse filtri konfiguratsioon

3.2 Front-end rakenduse arendamine



Joonis 3.11 Tundliku disaini näide

Front-end rakendus on realiseeritud üheleherakendusena (SPA – Single Page Application), kasutades JavaScript ja Vue.js.

Vue.js on kaasaegne JavaScript'i raamistik üheleherakenduste arendamise võimalusega. Vue.js kasutab komponendipõhist programmeerimismudelit, mis on kliendirakenduste arendamiseks väga kasulik. Lisaks on kasutatud Vuetify, mis on Vue.js põhjal loodud kasutajaliidese raamistik.

Front-end rakendus kasutab ka tundliku disaini, mis võimaldab rakendust kasutada mis tahes ekraanisuuruses või ekraani eraldusvõimega.

3.2.1 Registreerimine

```
export async function signup(username, password, firstname, lastname, email) {
  try {
    const response = await fetch("/api/auth/signup", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        firstname: firstname.value,
        lastname: lastname.value,
        username: username.value,
        email: email.value,
        password: password.value,
      }),
    });
    if (response.status == 201) {
      return true;
    } else {
      throw new Error("Signup Failed");
    }
  } catch (error) {
    console.error(error);
    return false;
  }
}
```

Joonis 3.12 Front-end registreerimise meetod

Registreerimiseks täidab kasutaja graafilise liidese kaudu vastava vormi. Pärast kinnitamisnupul klõpsamist kogub rakendus vormilt andmed ja edastab need lõpp-punkt API "/auth/signup" ja HTTP POST meetodiga (vt Joonis 3.12). Edukas registreerimine suunab kasutaja sisselogimislehele. Kui ilmneb registreerimise viga, kuvatakse kasutajale vastav teade.

```
export async function login(username, password) {
  try {
    const response = await fetch("/api/auth/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        username: username.value,
        password: password.value,
      }),
    });
    const responseData = await response.json();
    console.log(responseData);
    if (!responseData.responseCode || responseData.responseCode == "OK") {
      await saveUserDataToSessionStorage(responseData);
      return responseData;
    } else {
      throw new Error("Login Failed");
    }
  } catch (error) {
    console.error(error);
    return null;
  }
}
```

Joonis 3.13 Front-end login meetod

```
// Get a presigned PUT URL from the server
var getResponse = await fetch(
  "/api/users/" + userId + "/upload-profile-image",
  {
    method: "GET",
  }
);

console.debug(getResponse);
var putUrl = await getResponse.text();
console.debug("PUT url: " + putUrl);

// Use the presigned PUT URL to upload file
var putResponse = await fetch(putUrl, {
  method: "PUT",
  body: file,
  headers: {
    "Content-Type": "image/png",
  },
});
```

Joonis 3.14 Sisselogitud kasutaja võib oma kontopildi muuta

Autentimine toimub kasutajanime ja parooli saatmisega serverisse kontrollimiseks spetsiaalse API lõpp-punkti ja HTTP POST meetodiga (vt Joonis 3.13). Kui sisselogimine õnnestub, tagastatakse klientrakendusele JSON-objekt koos kasutaja informatsiooniga, mida edaspidi kastutatakse näiteks kasutaja avatari ja nime ning nende rollide kuvamiseks.

Rakenduse kasutamisel on kasutajatel juurdepääs funktsioonidele, mis vastavad kasutajakonto rollile. Oluline punkt on see, et selle lähenemisviisi puhul ei ole vaja kasutajarolle krüpteeritud kujul salvestada, kuna server haldab kasutajate päringuid seansside abil. Ründaja, kes asendab oma kasutajarolli, annab ainult visuaalse efekti, kuid ei luba tal täita volitamata taotlust.

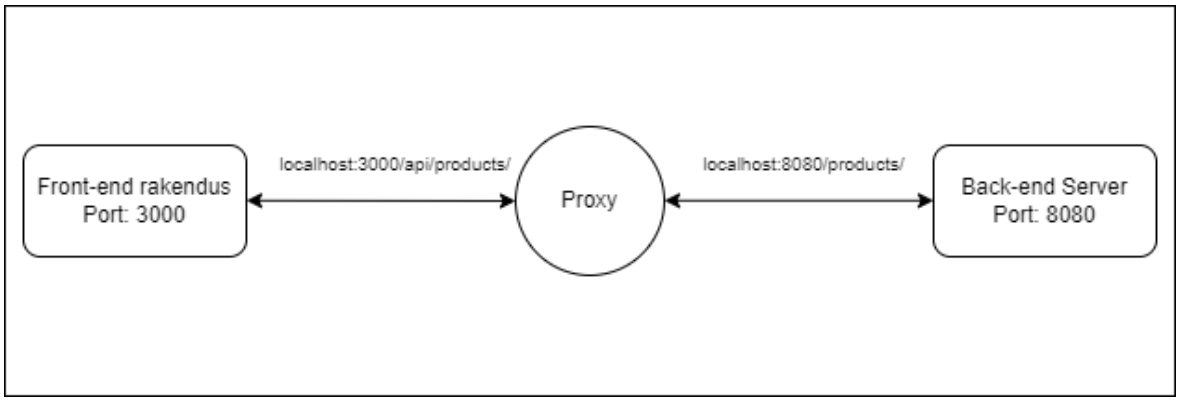
3.2.2 Puhvri seadistamine

Päringute serverisse ümbersuunamiseks kasutatakse puhverserverit. Arenduse käigus kasutatakse Vite arendusserverisse sisseehitatud puhverserverit. Rakenduse reaalsetes tingimustes käitamiseks peate seda lähenemisviisi muutma. Võimalusi on mitu: konfigureerida eraldi puhverserver, kasutada Axiose HTTP-klienti ja selle otse päringutesse sisseehitatud puhverservereid või saada päringuid otse serverisse. Väärib märkimist, et viimane valik võib põhjustada probleeme CORS-i seadistamisel.

```
server: {
  port: 3000,
  host: true,
  proxy: {
    "/api": {
      target: "http://localhost:8080",
      changeOrigin: true,
      rewrite: (path) => path.replace(/^\/api/, ""),
    },
  },
},
```

Joonis 3.15 Puhvri konfiguratsioon Vite arendusserveri jaoks

Kui rakendus esitab päringu URL-ile „localhost:3000/api/**“, saab puhverserver käsu suunata see päring enne määratud sihtmärgile, antud juhul URL-i „localhost:8080/**” (vt Joonis 3.16).



Joonis 3.16 Puhverserveri kasutamise diagramm

4. TULEMUS JA RAKENDUSE JUURUTAMINE

Viimasest peatükist saadud rakendus töötab kohalikus masinas edukalt. Järgmine samm on teha rakendus väljastpoolt ühendamiseks kättesaadavaks.

Esimene samm selle saavutamiseks on rakenduse konteinerisse paigutamine Dockeri abil. Selleks tuleb projekti kausta luua spetsiaalne fail nimega Dockerfile. See fail sisaldab samm-sammult juhiseid projekti koostamiseks ja käitamiseks. Sellise faili kirjutamise näide on näidatud joonisel 4.1, kuid see erineb erinevate rakenduste puhul. Toodud näites on Dockerfile jagatud 2 etapiks: esimene etapp ehitab rakenduse Maveni abil, mille tulemuseks on .jar laiendiga fail ja teine etapp käivitab loodud faili Java Runtime Environment versiooni 17 kasutades..

```
# Build
FROM maven:3.8.4-eclipse-temurin-17 AS build
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package

# Run
FROM eclipse-temurin:17-jre-alpine
WORKDIR /app
COPY --from=build /app/target/webstore-0.0.1-SNAPSHOT.jar ./backend.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "backend.jar"]
```

Joonis 4.1 Dockerfile'i näidis

Järgmine samm on Dockerfile'i kasutamine õige platvormiga konteineri pildi loomiseks. Selle prototüübi jaoks oli platvormiks Linux/amd64, millel põhineb Amazon Linux 2023 AMI operatsioonisüsteem (vt Joonis 4.3).

Kui on vaja käsurea abil selle arhitektuuriga konteineri luua, võib kasutada käsku "docker buildx build", määrates parameetri "--platform linux/amd64". [12].

Pärast seda saab loodud konteineri pildi Amazon ECR-i üles laadida. Selleks peate Amazon ECR-is looma privaatse hoidla ja kasutama konteineri pildi üleslaadimiseks Amazoni CLI-d. Detailsed juhised Amazon ECR-i kasutamiseks, mis on näidatud pildil 4.2, on osa ametlikust Amazon ECR-i dokumentatsioonist.

To push a Docker image to an Amazon ECR repository

The Amazon ECR repository must exist before you push the image. For more information, see Creating a private repository.

1. Authenticate your Docker client to the Amazon ECR registry to which you intend to push your image. Authentication tokens must be obtained for each registry used, and the tokens are valid for 12 hours. For more information, see Private registry authentication.

   To authenticate Docker to an Amazon ECR registry, run the **aws ecr get-login-password** command. When passing the authentication token to the **docker login** command, use the value `AWS` for the username and specify the Amazon ECR registry URI you want to authenticate to. If authenticating to multiple registries, you must repeat the command for each registry.

   > **Important**
   >
   > If you receive an error, install or upgrade to the latest version of the AWS CLI. For more information, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

   ```
   aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr
   ```

2. If your image repository doesn't exist in the registry you intend to push to yet, create it. For more information, see Creating a private repository.
3. Identify the local image to push. Run the **docker images** command to list the container images on your system.

   ```
   docker images
   ```

   You can identify an image with the `repository:tag` value or the image ID in the resulting command output.
4. Tag your image with the Amazon ECR registry, repository, and optional image tag name combination to use. The registry format is `aws_account_id.dkr.ecr.us-west-2.amazonaws.com`. The repository name should match the repository that you created for your image. If you omit the image tag, we assume that the tag is `latest`.

   The following example tags a local image with the ID `e9ae3c220b23` as `aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag`.

   ```
   docker tag e9ae3c220b23 aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
   ```

5. Push the image using the **docker push** command:

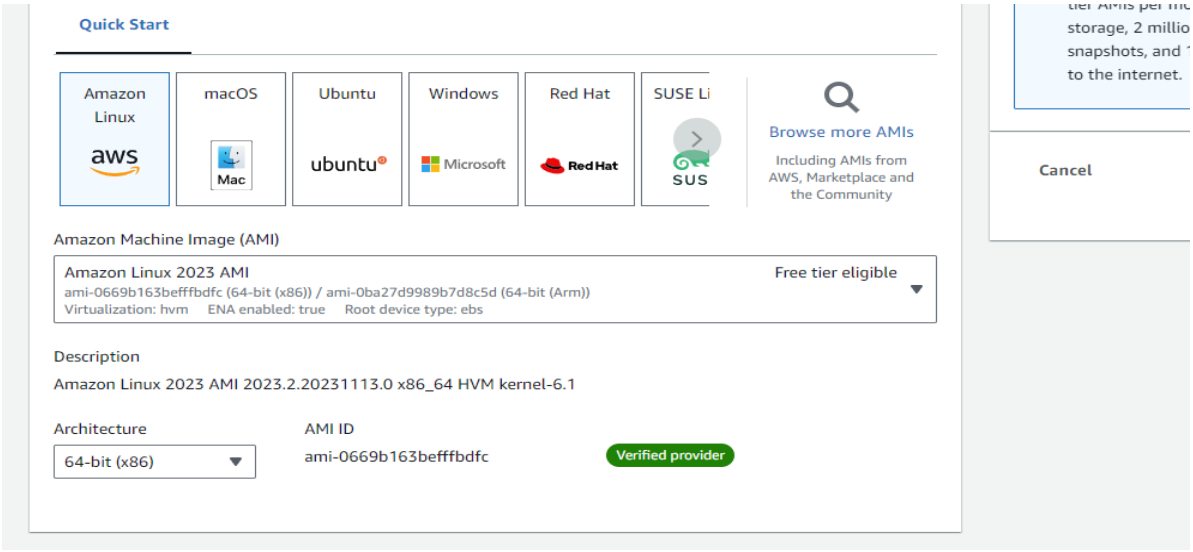
   ```
   docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
   ```

Joonis 4.2 Amazon ECR kasutamise juhend [13]

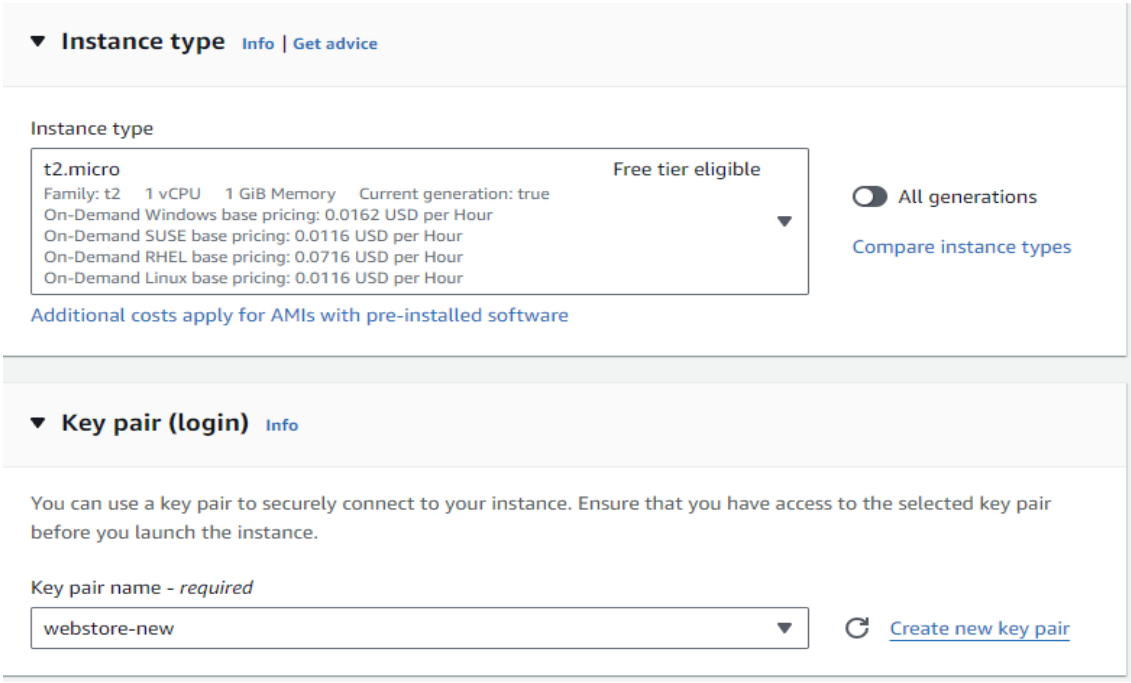
4.1 Amazon EC2 eksemplari käivitamine

See peatükk annab näite Amazon EC2 eksemplari käivitamisest. Kõigepealt on vaja määrata virtuaalse masina nime ja valida operatsioonisüsteemi. Autor valis Amazon Linux 2023 AMI 64-bit, kuna see kuulub tasuta kasutamise tingimuste alla (vt Joonis 4.3).

Järgmise sammuna tuleb valida eksemplari tüüp ja luua juurdepääsu jaoks võtmepaar (vt Joonis 4.4). Neid võtmeid tuleks hoida turvalises kohas, kuna neid läheb vaja virtuaalmasina konfigureerimiseks.



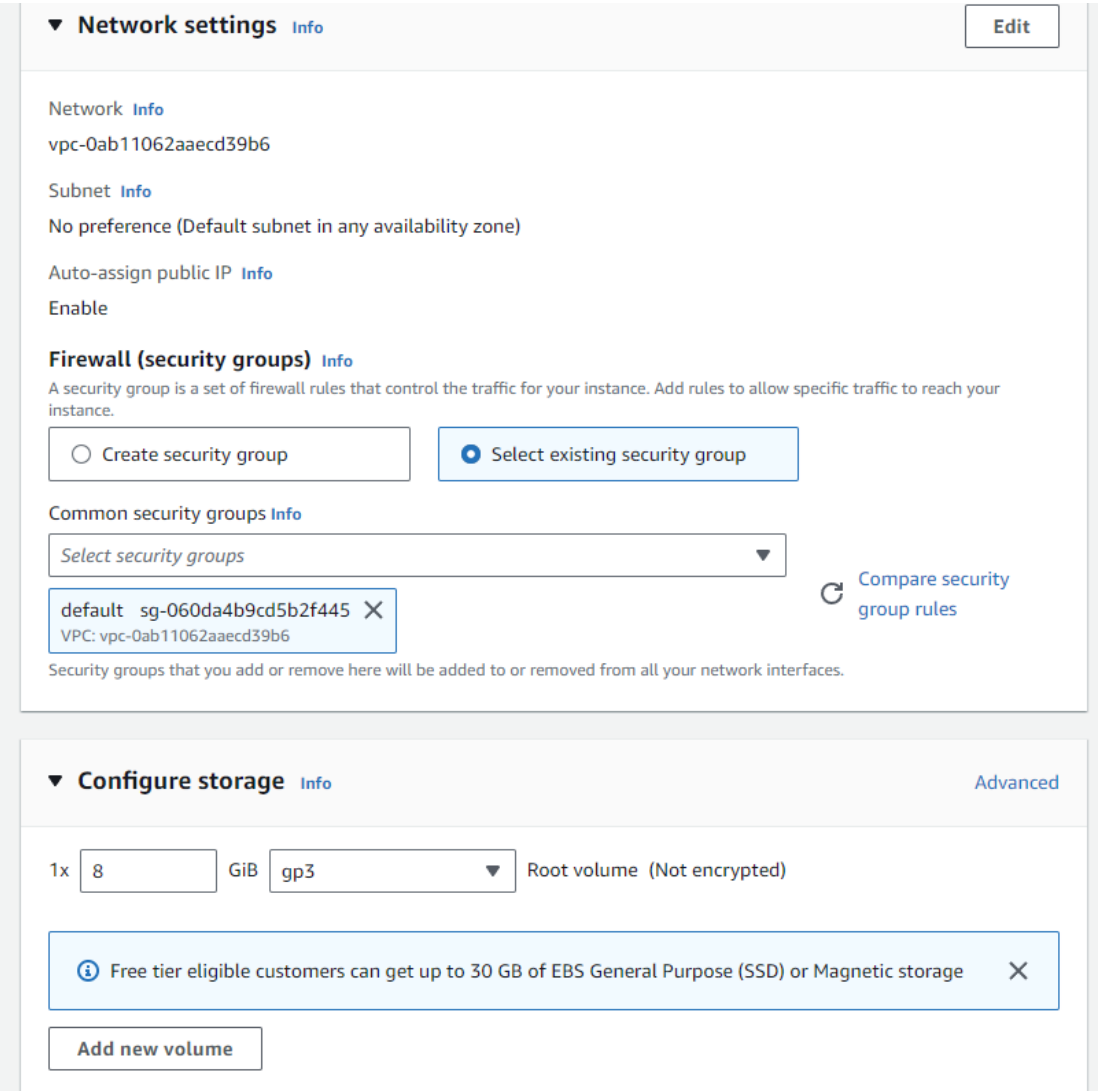
Joonis 4.3 Amazon EC2 virtuaalmasina seadistamine – 1



Joonis 4.4 Amazon EC2 virtuaalmasina seadistamine – 2

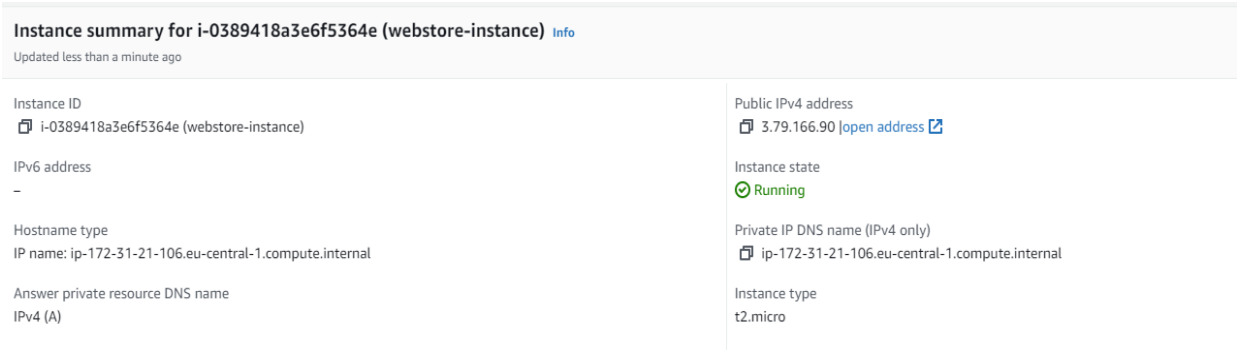
Järgmisena on vaja konfigureerida võrgusätted ja turvasätted. Siin on võimalik valida ühe eelnevalt loodud võrkude ja turvarühmade hulgast või luua uue. Erilist tähelepanu tuleks sellele pöörata juhul, kui arendatav rakendus peab olema avalikult kättesaadav.

Järgmine samm on virtuaalmasina kettaruumi valimine. Nagu pildilt näha, saab Amazon EC2 kasutada kuni 30 GB vaba kettaruumi (vt Joonis 4.5).



Joonis 4.5 Amazon EC2 virtuaalmasina seadistamine – 3

Pärast sisestatud sätete kinnitamist ja veidi ootamist käivitatakse EC2 eksemplar (vt Joonis 4.6). Selle eksemplari olekut näete veerus Instance State. Selle masinaga ühenduse loomiseks kasutatakse avalikku IPv4-aadressi.

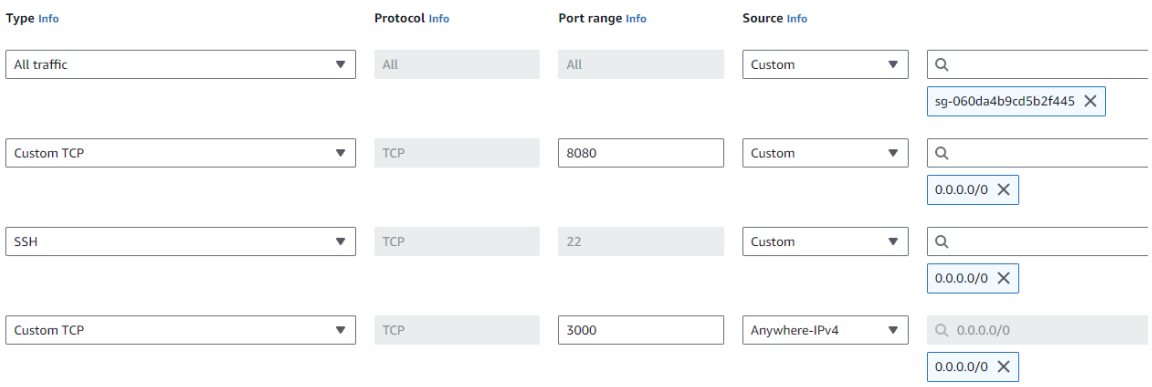


Joonis 4.6 Amazon EC2 virtuaalmasina staatus

4.2 Rakenduse juurutamine Amazon EC2 peal

Rakenduse juurutamiseks töötavale EC2 eksemplarile on soovitatav kasutada SSH-klienti.

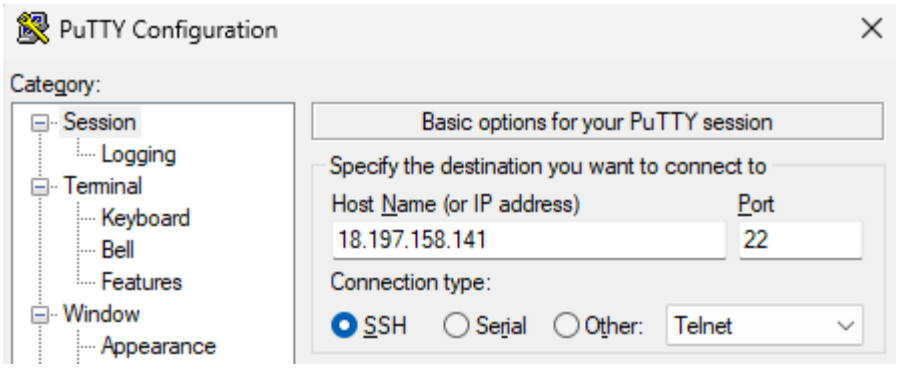
Tuleb veenduda, et virtuaalmasinale on juurdepääs pordi 22 kaudu. Selle jaoks tuleb minna Security Groups sätetesse ja valida Inbound Rules. On tähtis, et juurdepääsureeglid lubaksid ühendusi praeguse arvuti IP-aadressi jaoks. Sel juhul valis autor võimaluse ühenduda mis tahes IP-aadressilt (0.0.0.0/0).



Joonis 4.7 Turvalisuse reeglid

Pärast vajalike sätete tegemist on võimalik SSH abil ühenduse luua EC2 eksemplariga. Ühenduse loomiseks on vaja töötava virtuaalmasina avalikku IPv4-aadressi, samuti virtuaalmasina seadistamise käigus loodud võtmepaari.

Oluline märkus on see, et EC2 virtuaalmasinate IPv4-aadressid määratakse igal käivitamisel dünaamiliselt. Staatilise IP-aadressi määramine on eraldi teenus ja see ei kuulu Amazon Free Tier'i. [14].



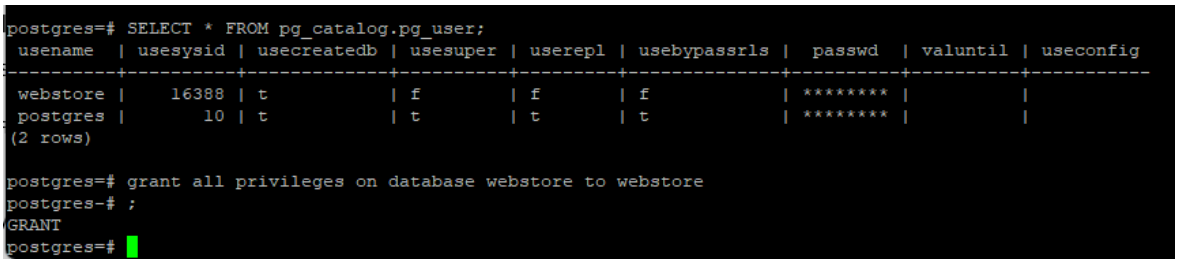
Joonis 4.8 SSH-ühenduse loomine EC2 eksemplariga PuTTY abil

Esmakordsel sisselogimisel tuleb kasutada valitud operatsioonisüsteemi vaikimisi kasutajanime. Amazon Linux 2023 puhul on see "ec2-user". Standardsete kasutajanimede täielik loetelu on saadaval ametlikus Amazon EC2 dokumentatsioonis. [15]

Joonis 4.9 Püstitatud SSH-ühendus

Eduka ühenduse näide on näidatud pildil 4.9. Pärast seda etappi tuleb läbida kõik ettevalmistavad sammud, näiteks: virtuaalmasina seadistamine vastavalt oma vajadustele, Dockeri, PostgreSQL-i ja muude tööriistade installimine.

Pärast PostgreSQL-i installimist tuleb luua andmebaasi kasutajat ja rakenduse jaoks andmebaasi vastavalt back-end rakenduse failis application.properties määratud parameetritele. Kasutajad peavad omama andmebaasiga töötamiseks vajalikke õigusi.



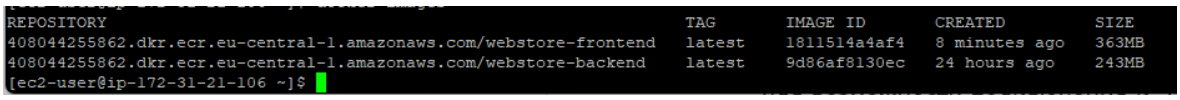
Joonis 4.10 PostgreSQL privileges

Järgmine samm on Amazon ECR-i autentimine, millele järgneb Docker Login, kasutades käsku «aws ecr get-login-password --region [Region] | docker login --username AWS --password-stdin [Account ID].dkr.ecr.[Region].amazonaws.com».

Pärast seda on saadaval käsk, mis on vajalik vajalike piltide allalaadimiseks Amazon ECR-ist:

«docker pull [Account ID].dkr.ecr.[Region].amazonaws.com/[Repository]:latest» [16]

EC2 virtuaalmasinasse alla laaditud konteineri kujutisi saab näha, käivitades käsu „docker images”.



Joonis 4.11 EC2 eksemplarile alla laaditud ECR-märgisega pildid

Konteineri käivitamiseks tuleb kasutada käsku «docker run -d [IMAGE_ID] --network=”host”». Parameeter “--network=”host” on vajalik rakendusele juurdepääsuks virtuaalmasina enda IP-aadressi abil. Pärast konteinerite käivitamist rakendusega tuleb kasutada käsku «docker logs [CONTAINER_NAME]» ja kontrollida, kas rakendus käivitus edukalt (vt joonis 4.13).

```
[ec2-user@ip-172-31-21-106 ~]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
f49848485009   1811514a4af4   "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes             competent_austin
251736d01396   9d86af8130ec   "java -jar backend.j…"   18 minutes ago   Up 18 minutes             kind_ritchie
[ec2-user@ip-172-31-21-106 ~]$
```

Joonis 4.12 Töötavate konteinerite nimekiri

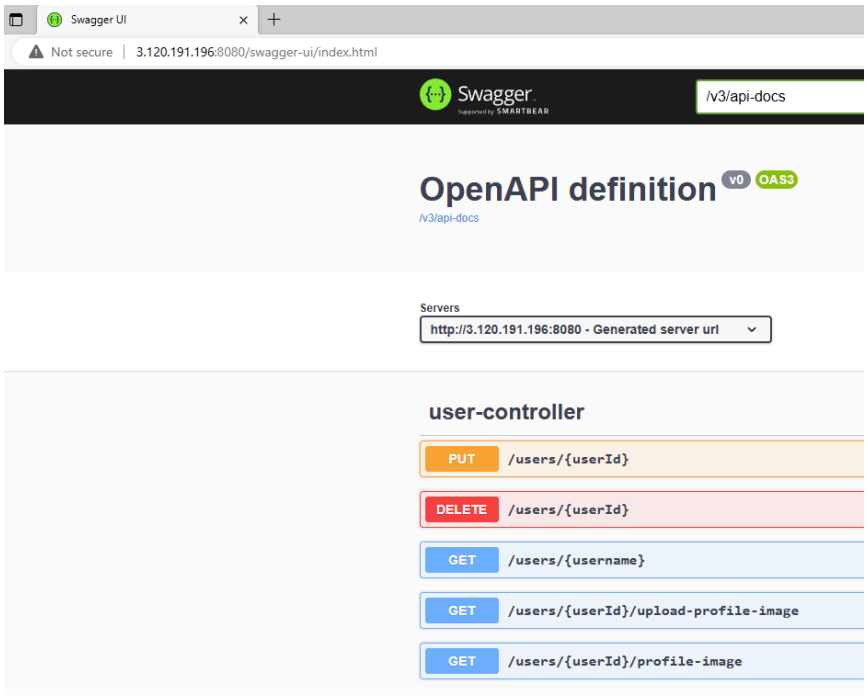
```
[ec2-user@ip-172-31-21-106 upgrade]$ docker run --network="host" -p 8080:8080 9d86af8130ec
WARNING: Published ports are discarded when using host network mode

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.4)

2023-12-07T14:18:48.200Z  INFO 1 --- [           main] com.shop.webstore.WebstoreApplication    : Starting WebstoreApplication v0.0.1-SNAPSHOT usi
g Java 17.0.9 with PID 1 (/app/backend.jar started by root in /app)
2023-12-07T14:18:48.214Z  INFO 1 --- [           main] com.shop.webstore.WebstoreApplication    : No active profile set, falling back to 1 default
profile: "default"
2023-12-07T14:18:50.815Z  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DE
AULT mode.
2023-12-07T14:18:50.982Z  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 155
s. Found 3 JPA repository interfaces.
2023-12-07T14:18:52.866Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-12-07T14:18:52.892Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-12-07T14:18:52.893Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.13]
2023-12-07T14:18:53.215Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationConte
t
2023-12-07T14:18:53.217Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization compl
ted in 4849 ms
```

Joonis 4.13 Back-end rakenduse konteineri logid

4.3 Rakenduse kasutamine ja testimine

Selles etapis on rakendus juurutatud ja kasutamiseks valmis. Rakendusele pääseb juurde virtuaalmasina IP-aadressi abil. Front-end rakendus töötab pordis 3000 ja back-end pordis 8080. Kasutades käsitsi testimist, tuleb veenduda, et kõik funktsioonid töötavad nii nagu vaja. Rakendusele pääseb ligi mis tahes veebibrauseriga, näited on toodud piltidel 4.14 ja 4.15.

Joonis 4.14 EC2 eksemplaris hostitud rakenduse kasutamine



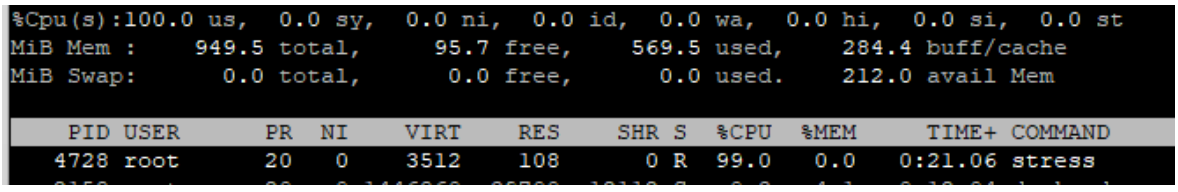
Joonis 4.15 EC2 eksemplaris hostitud rakenduse kasutamine - 2

4.4 Stress testimine

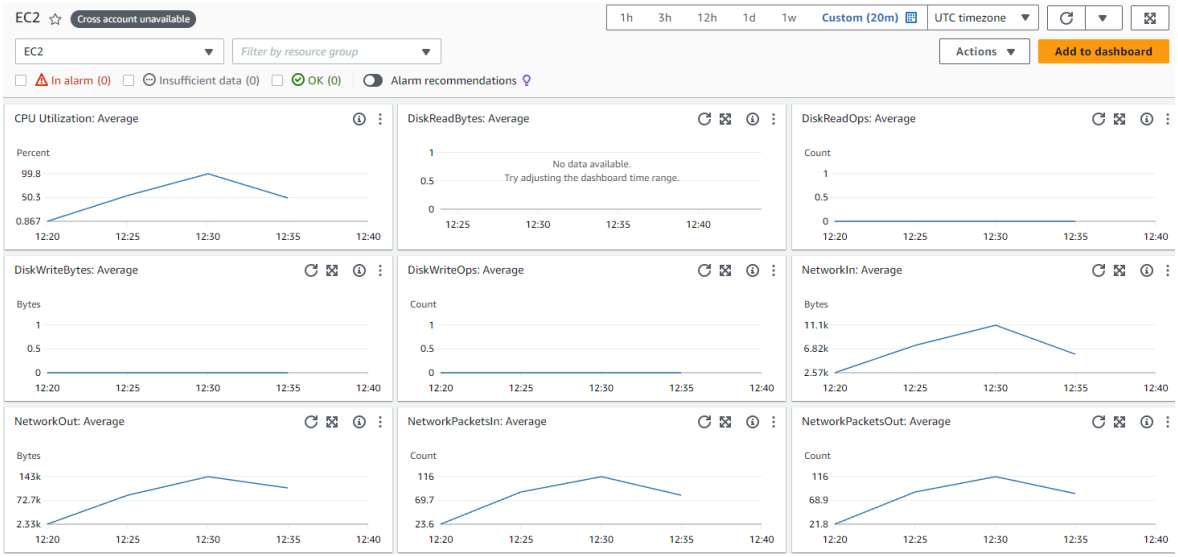
EC2 eksemplari koormuse all töötamise testimiseks viis autor läbi 600-sekundilise stressitesti, kasutades stressiutiliiti, mille abil luuakse süsteemile konfigureeritav koormus. Utiliit installitakse käsuga "sudo yum install stress".

Testi läbiviimisel saate määrata palju parameetreid. Sel juhul stressitest oli käivitatud kasutades käsku "sudo stress --cpu 1 --timeout 600 & top".

Tulemusi jälgiti Amazon CloudWatchi ressursside jälgimise süsteemi abil ja jooksvate protsesside jälgimisel top utiliidi abil. Testitulemused näitasid, et kuigi top näitas, et protsessori kasutus oli kogu testimisprotsessi jooksul 100%, kasvas Amazon CloudWatchi andmetel protsessori kasutus lineaarselt ja saavutas tipppunkti 100% alles testi lõpus, nagu on näha piltidelt 4.16 ja 4.17.



Joonis 4.16. Top utiliit annab teada protsessori kasutamisest

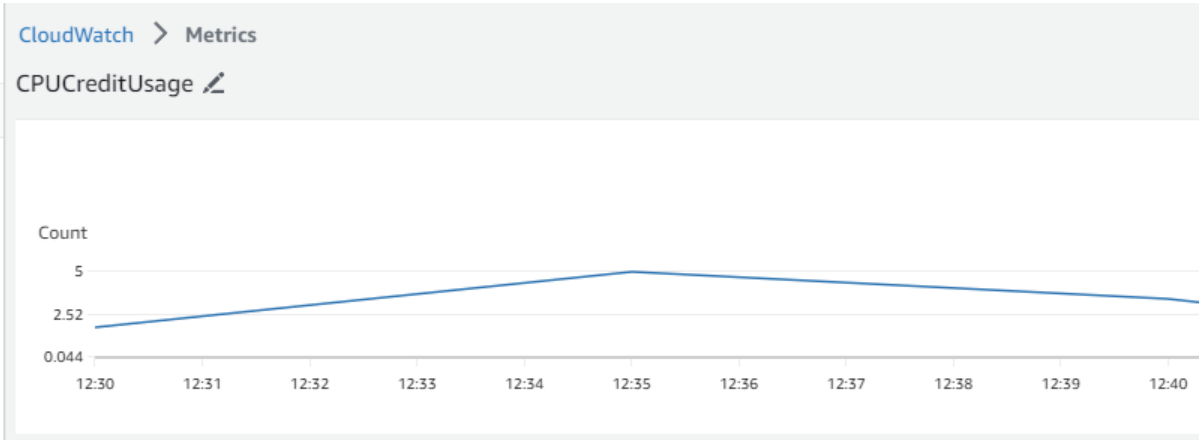


Joonis 4.17 EC2 stress testi tulemused (Amazon CloudWatch). Testimise aeg: 12:20 – 12:30.

Need tulemused on seletatavad sellega, et t2.micro tüüpi virtuaalmasinad, mis sisaldavad ka eelmises peatükis loodud Amazon EC2 eksemplari, kuuluvad Burstable tüüpi. Seda tüüpi virtuaalmasinatel on funktsioon, mis võimaldab neil koormuse all ajutiselt oma arvutusvõimsust suurendada.

Igal seda tüüpi masinal on teatud arv CPU krediite, mis aja jooksul kogunevad. Kasutades CPU krediite, saab virtuaalmasin ilma lisatasuta kasutada suuremat töötlemisvõimsust. [17]

See funktsioon võimaldas rakendust ilma viivitusteta kasutada isegi pideva koormuse korral.



Joonis 4.18 CPU krediite kasutamise logid (Amazon CloudWatch)

KOKKUVÕTE

Selle lõputöö kirjutamise käigus autor uuris erinevaid AWS-teenuseid, samuti võimalusi rakenduste konteinerisse paigutamiseks kiiremaks ja mugavamaks juurutamisprotsessiks.

Tulemuseks on mitme erineva AWS-i teenuse kasutusprofiili loomine ja rakenduse prototüübi loomine tasuta profiili abil. Prototüübiks sai ühes Amazon EC2 virtuaalmasinas töötava veebirakenduse loomine, et kuuluda Amazon Free Tier tingimuste alla.

Rakendus koosneb kahest osast – Vue.js baasil loodud kliendiosast ning Java ja Spring Frameworki baasil ehitatud serveriosast. Rakendus kasutab PostgreSQL andmebaasi ja Amazon S3 failisalvestusteenust. Rakendus käivitatakse Dockeri abil. Konteinerrakenduste pildid salvestatakse Amazon ECR-i abil. Rakendusele pääseb ligi EC2 virtuaalmasina IP-aadressi kaudu, kasutades porte 3000 ja 8080.

See lähenemisviis on käsitsi konfigureerimisel kõige nõudlikum, kuna see hõlmab kõigi vajalike tööriistade eelinstallimist, seadistamist ja seejärel konteinerite käivitamist käsurea abil.

Edasise arendusega on võimalik lülituda töö autori pakutud järgmisele AWS-teenuste kasutamise profiilile. Selleks tehakse ettepanek eraldada andmebaas, paigutades see Amazon RDS-i või eraldi Amazon EC2 eksemplari. Rakendus ise võib kas jätkata töötamist Amazon EC2 abil või vajaduse korral teisaldada mõnda rakendust töötavasse teenusesse, nagu Amazon Elastic Beanstalk või Amazon App Runner. See konfiguratsioon nõuab mõningaid finantskulusid, mida kompenseerib suurem autonoomia, skaleerimise võimalused ja töökindlus.

SUMMARY

The title of this thesis translates into English as „Web application development using AWS“. The author, Vjatšeslav Trelin, explores the application of Amazon Web Services in software development.

This work is divided into two bigger parts: an introduction to Amazon Web Services, and the development of a web application prototype using several of the services mentioned in the overview section.

As an introduction, author presents an overview of several established services provided by Amazon: Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, Amazon Elastic Compure Cloud, Amazon Fargate, Amazon Elastic Beanstalk, Amazon App Runner, Amazon Relational Database Service and Amazon Simple Storage Service. The overview consists of an explanation of their main features and possible use cases. It continues with the author providing several usage profiles, that combine many of those services together and providing a sample for the architecture. The profiles were created and ranked based on several criterias, such as usage price and scope.

In the following part, author proceeds to use the first suggested AWS profile to build and host a web application. The application prototype is separated into two main parts, the first one being a backend server built with Java and Spring Framework. The server provides an API for client and server to communicate, while also establishing a data access layer, which is used for storing application data in a PostgreSQL database. The server also provides presigned URLs for Amazon S3 data storage, which can be later used by the client side to download and upload images.

The second part is a frontend client, built with JavaScript, Vue.js and Vuetify. The client provides an interactive user interface for the end user. It communicates with the server using an API provided by the server side.

As a result, the web application was successfuly created, divided into separate Docker containers, and was deployed on a single Amazon EC2 instance. The EC2 instance is configured to be accessable via ports 3000, 8080 and 22. It allows the end user to access the frontend application, the server directly, while also providing a way control the virtual machine using SSH.

In the future, this setup can be further improved by decoupling a database, switching to any automated AWS application hosting services, by extending the prototype application’s functionality and creating a CI/CD pipeline for quick application deployment.

KASUTATUD KIRJANDUSE LOETELU

1. Felix Richter. Amazon Maintains Lead in the Cloud Market. [Online] https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (11.10.2023)
2. Overview of Amazon Web Services [Online] https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html (11.10.2023)
3. AWS Elastic Beanstalk Features [Online] https://aws.amazon.com/elasticbeanstalk/details/ (20.11.2023)
4. Amazon ECS Features [Online] https://aws.amazon.com/ecs/features/ (17.12.2023)
5. Amazon EKS Features [Online] https://aws.amazon.com/eks/features/ (17.10.2023)
6. AWS Fargate Serverless compute for containers [Online] https://aws.amazon.com/fargate/ (17.10.2023)
7. Amazon S3 Storage Classes - Performance across the S3 storage classes [Online] https://aws.amazon.com/s3/storage-classes/?nc=sn&loc=3 (27.12.2023)
8. Amazon Route 53 [Online] https://aws.amazon.com/route53/?nc2=type_a (27.12.2023)
9. AWS Free Tier [Online] https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=tier%23always-free%7Ctier%2312monthsfree&awsf.Free%20Tier%20Categories=*all (20.11.2023)
10. Amazon ElastiCache [Online] https://aws.amazon.com/elasticache/?nc2=type_a (27.12.2023)
11. Developer Guide - AWS SDK for Java 1.x [Online] https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/welcome.html (01.12.2023)
12. docker buildx build [Online] https://docs.docker.com/engine/reference/commandline/buildx_build/ (01.12.2023)

13. Pushing a Docker image [Online] https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-push-ecr-image.html (27.12.2023)

14. Elastic IP addresses [Online] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html#eip-pricing (05.12.2023)

15. Manage users on your Linux instance [Online] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-users.html (05.12.2023)

16. Pulling an image [Online] https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-pull-ecr-image.html (05.12.2023)

17. Amazon EC2 Instance Type Details [Online] https://aws.amazon.com/ec2/instance-types/#burst (08.12.2023)