

# EUV Big Data Challenge

## Proof Of Concept

### Slimane El Boujadaini

## Graduaat Systeem- en Netwerkbeheer

Eurovision Voting Simulation

Creatie: 08-05-2025

Laatste wijziging: 11-06-2025

Auteur: Slimane El Boujadaini

[Voorwoord](#)

[Doelstelling](#)

[Specificatie](#)

[Fase 2: Docker op NAS \(Gestopt\)](#)

[Fase 3: Cloud-integratie](#)

[Fase 4: Rclone Synchronisatie](#)

[Fase 5: Git Versiebeheer](#)

[Technische Infrastructuur Overzicht](#)

[Systeemvereisten](#)

[Data Flow Pipeline](#)

[Infrastructuur setup](#)

[Download Links](#)

[Vereisten](#)

[How To](#)

[Installatiehandleiding: VirtualBox](#)

## Installatiehandleiding: CentOS VM aanmaken in VirtualBox

Schaalbaarheid

Proof Of Concept

Handleiding Overzicht

Project Setup en Configuratie

Uitvoering

Resultaat

Generator Container - Stemdata

Aggregator Container - Apache Spark Verwerking

Analyzer Container - Finale Ranking

Docker Compose Orchestrator

Volledige Pipeline Uitvoering

Data Flow en Bestandsstructuur

Troubleshooting en Debugging

Gedistribueerd Concept

Full Installation + Rclone

## **Voorwoord**

Dit IT-project werd uitgevoerd in het kader van mijn opleiding Graduaat Systeem- en Netwerkbeheer aan Odisee en vormt een belangrijke mijlpaal in mijn academische traject naar het behalen van mijn diploma en het bewijzen van mijn “technical en soft” skills.

Het project bood mij de unieke kans om als een zelfsturende professional geavanceerde technologieën en methodieken te leren kennen en toe te passen rond Big Data. Waaronder containerisatie met Docker, gedistribueerde data verwerking, en integratie met de Google Drive API service voor veilige en efficiënte cloudbaseerde dataopslag.

Tijdens de uitvoering heb ik zowel zelfstandig als in teamverband gewerkt. Samenwerkingen met medestudenten zoals Kurt Van Rymenant rond de API-architectuur en Ilias Briami betreffende het delen

en valideren van stemgegevens van verschillende landen hebben mijn project aanzienlijk verrijkt en nieuwe perspectieven geboden.

Hoewel het project aanvankelijk eenvoudig leek, bracht het diverse technische uitdagingen met zich mee. Het opzetten van de infrastructuur - van lokale virtuele omgevingen tot Docker containers met externe cloud storage - vereist grondige analyse en probleemoplossing. Deze uitdagingen hebben mijn technische vaardigheden en probleemoplossend vermogen aanzienlijk versterkt.

Het hoofddoel van het project was het ontwikkelen van een geautomatiseerd systeem voor het verzamelen, verwerken en analyseren van Eurovision Songfestival stemmen, met als eindresultaat de bepaling van een winnaar door middel van geavanceerde data-analyse om zo snel mogelijk een groot aantal stemmingen te verwerken, te behandelen en de integriteit van de resultaat te garanderen. Dit project heeft mij niet alleen technisch doen groeien, maar ook mijn vaardigheden in samenwerking, zelfsturing en professionele communicatie ontwikkeld. De feedbacksessies hebben mij geholpen om mijn project bij te sturen en continue verbetering na te streven.

Tot slot wil ik mijn medestudenten bedanken voor hun waardevolle input en constructieve samenwerking, en in het bijzonder Yvan Rooseleer voor zijn deskundige begeleiding en inzichten die zowel dit project als mijn professionele ontwikkeling hebben verrijkt.

## **Doelstelling**

In het kader van het Eurovision Songfestival Contest kregen we de opdracht van onze leerkracht Yvan Rooseleer om een betrouwbaar en schaalbaar stemsysteem op te zetten, te configureren, te testen en te

valideren. Het doel van dit project is om de stemmen per land binnen een specifieke tijdsperiode efficiënt te verzamelen, te verwerken en te tellen.

Mijn proof of concept verwerkt momenteel 100.000 stemmen per land. Aanvankelijk richtte ik me op Italië als testland, maar het project is uitgebreid naar meerdere landen om de validiteit en robuustheid van het systeem te waarborgen.

Meer informatie vind je in het document van onze stakeholder: [BiGDaTA EUV Briefing](#)

## Specificatie

Fase 1: Lokale Implementatie

De eerste fase werd volledig lokaal uitgebouwd met een eenvoudige architectuur:

- Platform: Linux VM als basis
- Gegevensverwerking: Data werd lokaal gelezen en geschreven
- Uitvoering: Scripts werden lokaal uitgevoerd

```
aggregator/:
total 8
-rw-r--r--. 1 slim slim 1503 Jun 11 18:26 aggregate_votes.py
-rw-r--r--. 1 slim slim 160 Jun 11 18:27 Dockerfile

analyzer/:
total 8
-rw-r--r--. 1 slim slim 3416 Jun 11 18:35 final_ranking.py
-rw-r--r--. 1 slim slim 170 Jun 11 18:36 Dockerfile

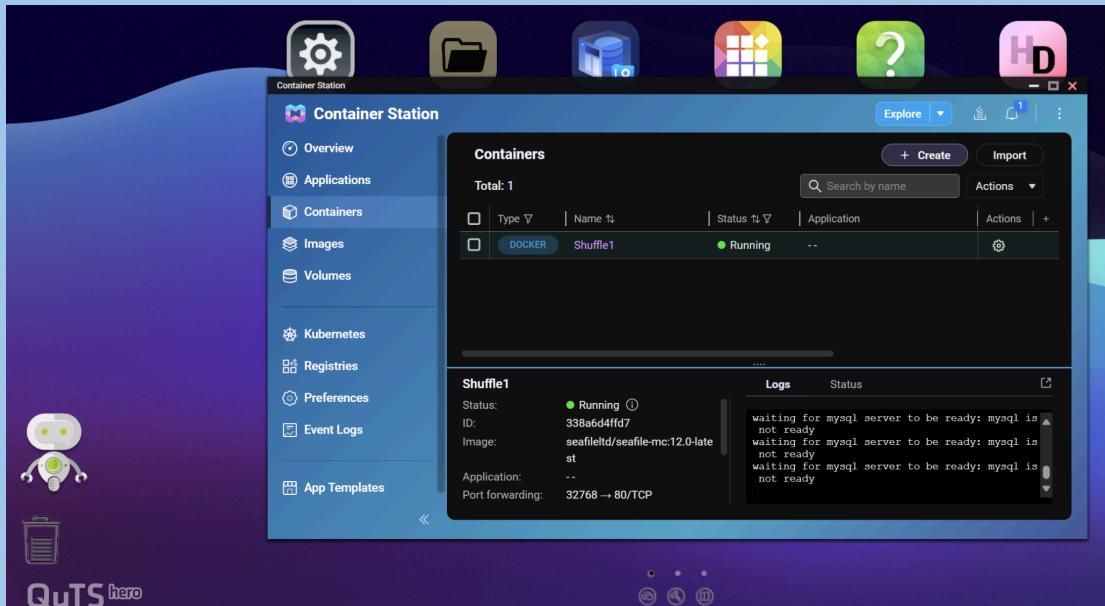
generator/:
total 8
-rw-r--r--. 1 slim docker 2849 Jun 11 19:01 generate_votes.py
-rw-r--r--. 1 slim docker 214 Jun 11 19:02 Dockerfile
```

folder structuur lokaal op het linux Centos VM

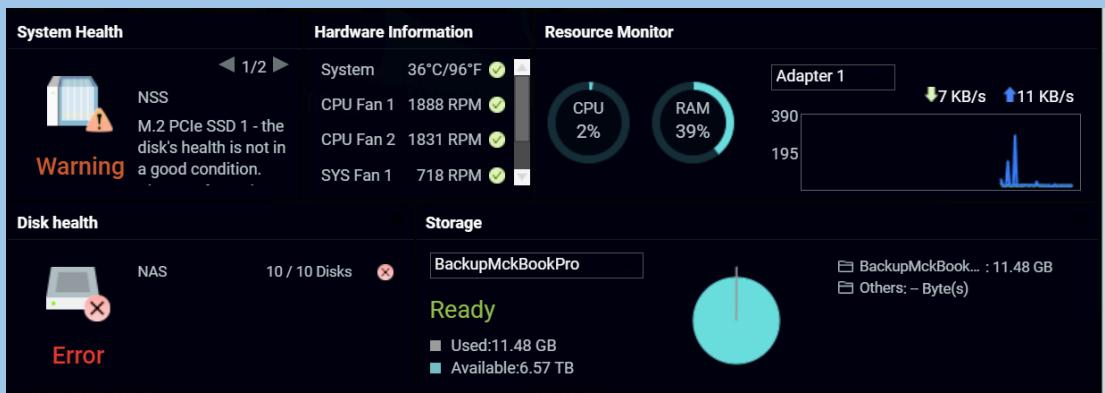
## Fase 2: Docker op NAS (Gestopt)

De tweede fase richtte zich op containerisatie in een cloud-omgeving:

- Technologie: Docker-implementatie op een NAS-systeem
- Doelstelling: Volledig cloud-gebaseerde oplossing
- Resultaat: Project werd stopgezet vanwege technische problemen, waaronder schijf- en netwerkproblemen



Een dockercontainer draaiend op een NAS server via container station.



Foutmelding gerelateerd aan het Hardware die de performantie impacteren

## Fase 3: Cloud-integratie

De derde fase breidde de oorspronkelijke lokale opzet uit naar de cloud:

- Cloudprovider: Google Drive als opslagoplossing
- Beheertools: Google Console voor projectbeheer
- Authenticatie: Service accounts met gedeelde private sleutels tussen hosts en cloud storage

google drive - private key and folder ID

## Fase 4: Rclone Synchronisatie

**De vierde fase implementeerde een geautomatiseerde synchronisatiestrategie1:**

- Technologie: Rclone voor gegevenssynchronisatie
  - Automatisering: Crontab voor geplande synchronisatie
  - Frequentie: Lokale folders worden elke minuut gesynchroniseerd met Google Drive
  - Voordeel: Oplossing voor NAS-problematiek door lokale opslag te behouden
  - Werkwijze: Alle data wordt lokaal geschreven en opgeslagen, vervolgens automatisch online gesynchroniseerd voor cloudbeschikbaarheid

```
* * * * * /usr/bin/rclone sync /home/slim/EUV-0disee euvdrive: >> /home/slim/cron_rclone.log 2>&1
```

## crontab content

```
[slim@vbox EUV-Odisee]$ rclone version
rclone v1.69.3
- os/version: centos 10 (64 bit)
- os/kernel: 6.12.0-89.el10.x86_64 (x86_64)
- os/type: linux
- os/arch: amd64
- go/version: go1.24.3
- go/linking: static
- go/tags: none
[slim@vbox EUV-Odisee]$ echo $PATH
/home/slim/google-cloud-sdk/bin:/home/slim/.local/bin:/home/slim/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[slim@vbox EUV-Odisee]$ rclone config
Current remotes:

Name          Type
=====        ====
euvdrive      drive

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> ■
```

Rclone versie en configuratie

## Fase 5: Git Versiebeheer

De vijfde en laatste fase focuste op professioneel versiebeheer:

- Platform: Git voor versiebeheer
- Inhoud: Synchronisatie van documentatie, resultaten en scripts
- Structuur: Georganiseerde workflow voor projectoutputs

Het belangrijkste verschil tussen fase 3 en 4 is de hybride aanpak: waar fase 3 direct met de cloud werkte, combineert fase 4 lokale betrouwbaarheid met cloud-toegankelijkheid door middel van geautomatiseerde synchronisatie

## Technische Infrastructuur Overzicht

Component	Functionaliteit	Technologie	Input	Output
Generator	Genereert realistische stem data per land (landcode,mobiel nummer, songnummer, timestamp)	Python, Docker	Gebruikersinput of geautomatiseerde parameters	italy_votes.txt
Aggregator	Verwerkt stemdata met PySpark en transformaties,	PySpark, Python, Docker	Tekstbestanden per land (*.votes.txt)	reduced_votes.json

	implementeert MapReduce-logica			
Analyzer	Analyseert geaggregeerde data, berekent eindklassement en bepaalt winnaar	Pandas, NumPy, Python, Docker	Geaggregeerde JSON-bestanden	Ranking-bestanden en visualisaties

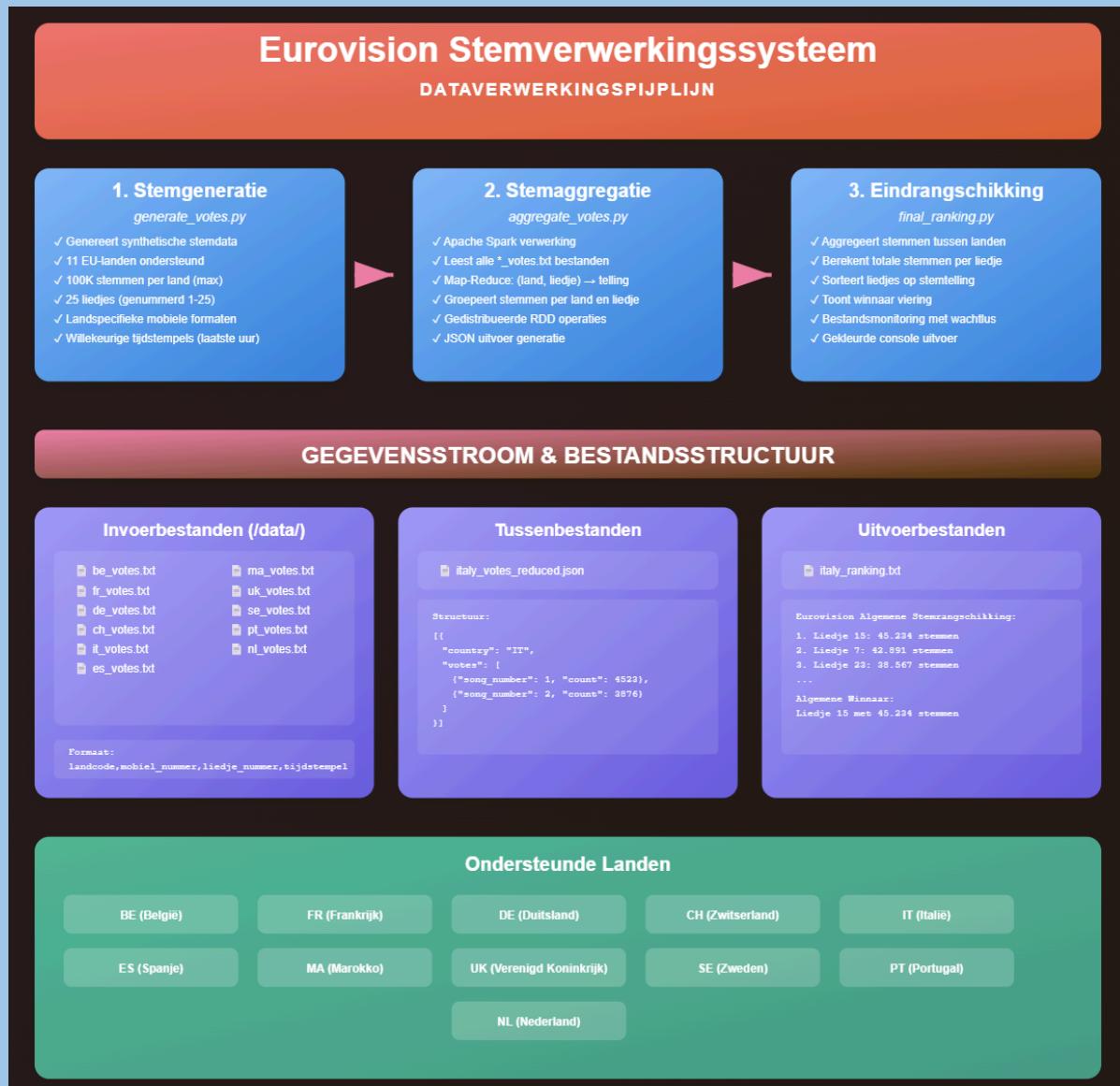
## Systeemvereisten

Categorie	Specificatie	Versie/Capaciteit
Containerisatie	Docker	20.10+
Orchestrator	Docker Compose	v2+
Runtime	Python	3.8+
Data Processing	pandas, numpy, pyspark	Latest stable
Geheugen	RAM	Minimaal 2GB
Opslag	Vrije schijfruimte	Minimaal 10GB

Besturingssysteem	OS Ondersteuning	Linux (Centos)
-------------------	------------------	----------------

### Data Flow Pipeline

Stap	Container	Proces	Data Transformatie
1	Generator	Data Generatie	Parameters → Stembestanden
2	Aggregator	MapReduce Verwerking	Stembestanden → JSON Aggregatie
3	Analyzer	Resultaat Analyse	JSON Data → Rankings & Visualisaties



Figuur 4: Eurovision Song Contest technical pipeline.

## Infrastructuur setup

### Download Links

Software	Beschrijving	Download Link
CentOS Stream 10	Linux distributie voor serveromgeving	<a href="#">Download</a>

Apache Hadoop	Framework voor gedistribueerde opslag en verwerking	<a href="#">Download</a>
Apache Spark 3.5.5	Gedistribueerd data processing framework	<a href="#">Download</a>
Python 3.11	Programmeertaal versie 3.11	<a href="#">Download</a>
Python Numpy	Python bibliotheek voor numerieke berekeningen	<a href="#">Download</a>
Python Pandas	Data manipulation en analyse bibliotheek	<a href="#">Download</a>
Docker	Containerisatie platform	<a href="#">Download</a>
Docker Compose	Multi-container Docker applicaties	<a href="#">Download</a>
Docker Hub	Docker Catalog	<a href="#">Download</a>
OpenJDK 17	Java Development Kit voor Java 17	<a href="#">Download</a>
pip	Python package manager	<a href="#">Download</a>
Google Cloud SDK	Command line tools voor Google Cloud	<a href="#">Download</a>

Google Client API	Python bibliotheek voor Google APIs	<a href="#">Install via pip</a>
Google Auth Libraries	Authenticatie voor Google services	<a href="#">Install via pip</a>
PySpark	Python API voor Apache Spark	<a href="#">Install via pip</a>
Git	Versiebeheer systeem	<a href="#">Download</a>
VirtualBox	Virtualisatie platform (voor VM setup)	<a href="#">Download</a>

## Vereisten

Component	Versie	Doel
VirtualBox	Nieuwste	VM Hypervisor
CentOS Stream 10 ISO	Nieuwste	Besturingssysteem
Systeem RAM	16GB+	Host beide VM's

Opslag	100GB+	VM opslagruimte
Applicatie's	Docker,Docker Compose,Python,Pandas,Numpy,Pyspark	Deze applicaties zijn de minimum vereiste applicaties om je script lokaal te testen

## How To

### Installatiehandleiding: VirtualBox

#### Stap 1: Download VirtualBox

1. Ga naar de officiële website:  
<https://www.virtualbox.org>
2. Klik op Downloads.
3. Kies het juiste installatiebestand voor jouw besturingssysteem:
  - Windows
  - macOS
  - Linux distributions (zoals Ubuntu, Fedora, etc.)

---

## Stap 2: Installeer VirtualBox

1. Open het gedownloade installatiebestand.
2. Volg de installatiewizard:
  - Klik op Next om door te gaan.
  - Kies optioneel welke componenten je wil installeren.
  - Klik op Install.
3. Bevestig eventuele waarschuwingen van Windows/macOS.
4. Wacht tot de installatie voltooid is.
5. Klik op Finish om VirtualBox te starten.

---

## Stap 3 (optioneel): Installeer VirtualBox Extension Pack

1. Download het Extension Pack van dezelfde downloadpagina.
2. Dubbelklik op het .vbox-extpack bestand.
3. Volg de prompts om te installeren.
4. Hiermee voeg je ondersteuning toe voor:
  - USB 2.0/3.0 apparaten
  - Remote Desktop Protocol (RDP)

- Versleutelde virtuele schijven
- 

## Controle

- Start VirtualBox via het startmenu of bureaublad.
- Je bent nu klaar om het eerste VM aan te maken!

## Installatiehandleiding: CentOS VM aanmaken in VirtualBox

### Stap 1: Download CentOS ISO

1. Ga naar de officiële CentOS Stream downloadpagina:  
<https://www.centos.org/download/>
  2. Kies:
    - CentOS Stream 10 ISO of
    - CentOS Stream 9 ISO (indien nodig voor compatibiliteit)
  3. Download de DVD ISO (of de minimal ISO als je zelf alles wil instellen).
- 

### Stap 2: Nieuwe VM aanmaken in VirtualBox

1. Open VirtualBox.
2. Klik op New of Nieuwe.
3. Geef de VM een naam, bijv. CentosDockerShuffle1

4. Kies:

- Type: Linux
- Version: Red Hat (64-bit)

5. Klik op Next / Volgende.

---

### Stap 3: Geheugen en harde schijf instellen

1. Memory Size: Kies minstens 2048 MB (aanbevolen 4096 MB voor betere prestaties).
2. Create a virtual hard disk now → klik op Create.

#### Schijfinstellingen:

- VDI (VirtualBox Disk Image)
- Dynamically allocated
- Grootte: 20 GB of meer (afhankelijk van je behoeften)

#### Klik op Create.

---

### Stap 4: ISO-bestand koppelen (opstartimage)

1. Selecteer de VM → Klik op Settings / Instellingen.
2. Ga naar het tabblad Storage.

3. Onder Controller: IDE, klik op de lege cd-rom → rechts klik op het cd-icoon → Choose a disk file.
  4. Selecteer de eerder gedownloade CentOS ISO.
  5. Bevestig met OK.
- 

## Stap 5: Start de VM en installeer CentOS

1. Selecteer de VM en klik op Start.
  2. De installatie van CentOS begint vanaf het ISO-bestand.
  3. Volg de stappen op het scherm:
    - Taal selecteren
    - Installatiebron is automatisch ingesteld (ISO)
    - Doelschijf kiezen (gebruik volledige schijf)
    - Netwerkinstellingen activeren (optioneel)
    - Rootwachtwoord instellen
    - Gebruiker aanmaken (optioneel)
  4. Klik op Begin installatie.
  5. Herstart de VM na installatie en verwijder de ISO uit de virtuele drive.
-

## Stap 6: Post-installatie

1. Log in met je gebruikersnaam of als root.
2. Test je installatie manueel of met behulp van een script zoals prerequisite.sh (Date,Timezone,Netwerk, enzo...)

```
[slim@vbox ~]$ whoami
slim
[slim@vbox ~]$ pwd
/home/slim
[slim@vbox ~]$ date
Wed Jun 11 02:35:33 AM CEST 2025
[slim@vbox ~]$ ls -lrt
total 0
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Downloads
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Desktop
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Templates
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Public
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Documents
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Videos
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Pictures
drwxr-xr-x. 2 slim slim      6 May 30 23:12 Music
drwxr-xr-x. 3 slim docker   25 May 31 00:27 Euvsong
drwxr-xr-x. 4 slim slim     64 Jun  4 00:50 seafile
drwxr-xr-x. 3 slim slim     21 Jun  8 09:56 Odisee
drwxr-xr-x. 2 slim slim     76 Jun  8 12:40 scripts
drwxr-xr-x. 7 slim slim    108 Jun  8 13:25 euv_project
drwxr-xr-x. 3 root root    16 Jun  8 16:04 path
drwxr-xr-x. 2 slim slim    65 Jun  8 16:07 testproject
drwxr-xr-x. 6 slim docker  125 Jun  9 00:05 eurovision-poc
[slim@vbox ~]$ cd scripts/
[slim@vbox scripts]$ ls
prerequisite.sh  setupinfracloud.py  setupinfra.py
[slim@vbox scripts]$ sh prerequisite.sh
==== Eurovision Pipeline Prerequisite Checker ====

Detected OS: CentOS Stream 10 (Coughlan)
[OK] Python 3 gevonden: Python 3.12.10
[OK] Docker gevonden: Docker version 28.2.2, build e6534b4
[OK] Docker CLI gevonden.
[OK] Docker Compose (v2 plugin) gevonden: Docker Compose version v2.36.2

==== Samenvatting ====
Besturingssysteem: CentOS Stream 10 (Coughlan)
Python 3: JA
Docker: JA
Docker Compose: JA

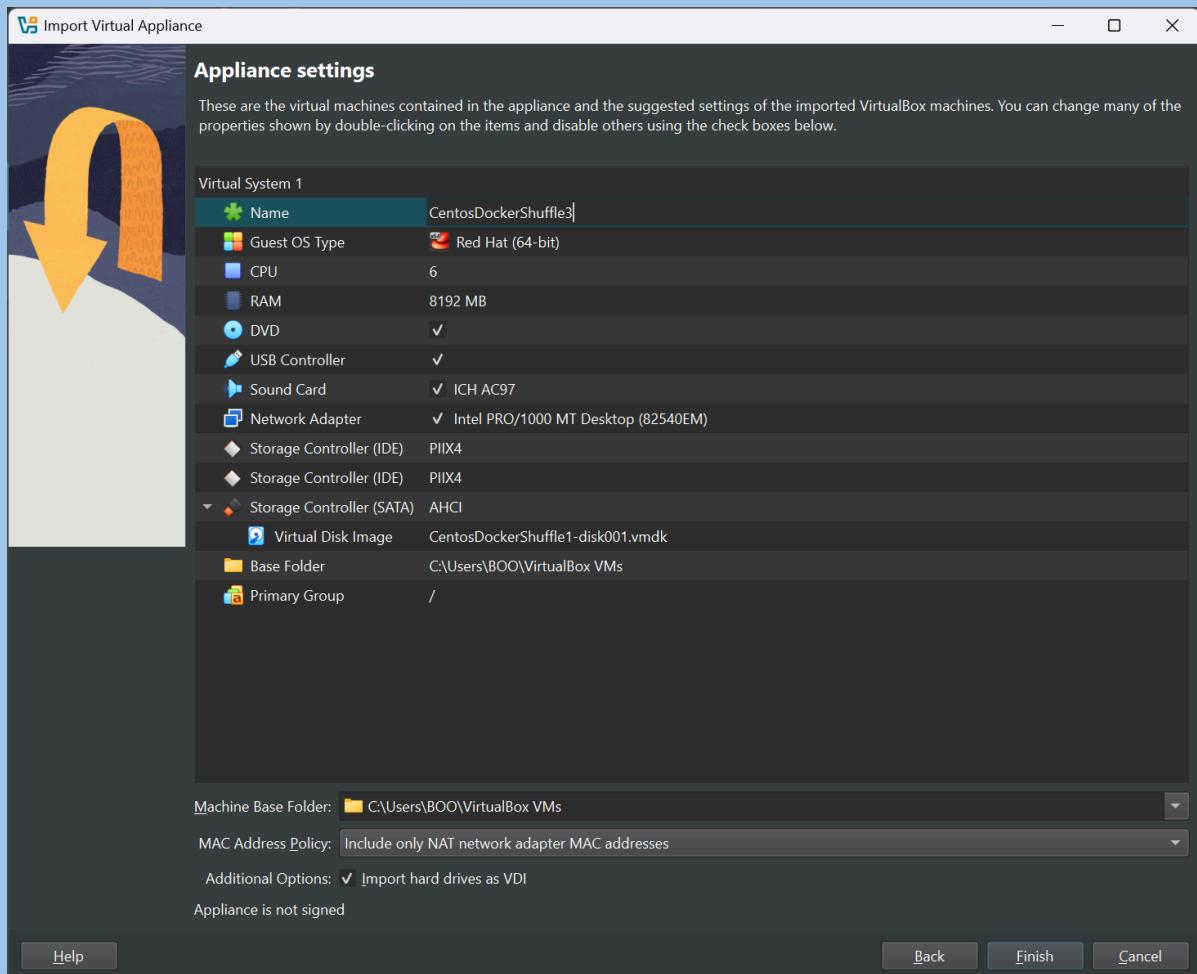
Volg de bovenstaande instructies om ontbrekende vereisten te installeren.
Na installatie kun je verder met 'python3 setupinfra.py'.
[slim@vbox scripts]$ █
```

Figuur 5: Script om te controleren dat de prerequisite voldaan zijn

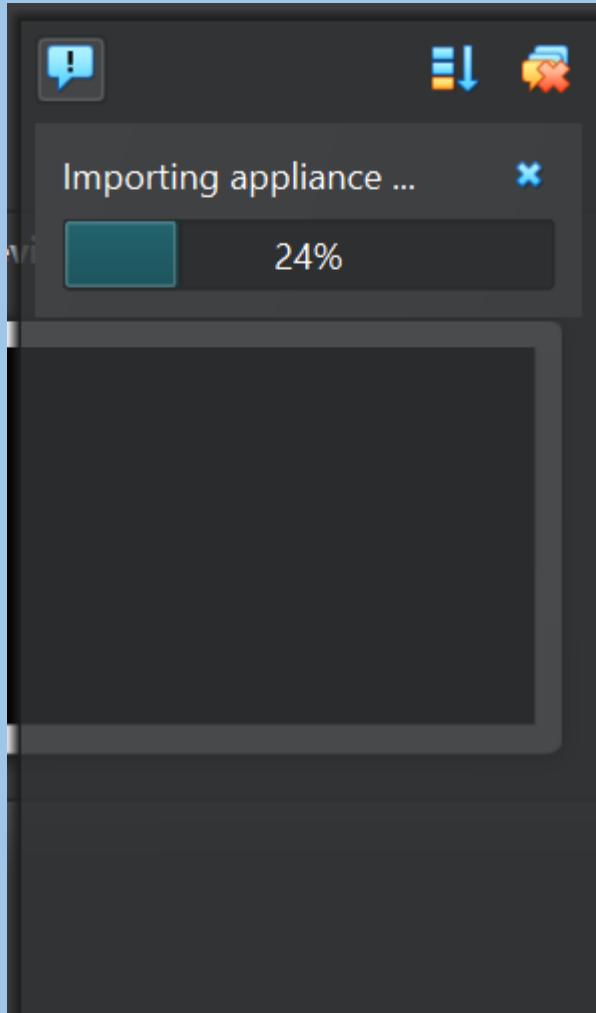
## Schaalbaarheid

Voor snelle schaalbaarheid bij verhoogde data volumes hebben we vooraf geconfigureerde VM-appliances ontwikkeld en geëxporteerd, deze is beschikbaar in de cloud onder het [infra-setup](#) folder . Deze kant-en-klare appliances kunnen onmiddellijk worden geïmporteerd en geactiveerd om de verwerkingscapaciteit van shuffle 1 (aggregator) en shuffle 2 (analyzer) uit te breiden binnen onze gedistribueerde infrastructuur. Deze appliancestrategie maakt horizontale schaalbaarheid mogelijk zonder tijdrovende configuratie, wat essentieel is voor het Eurovision stemsysteem bij piekbelastingen.

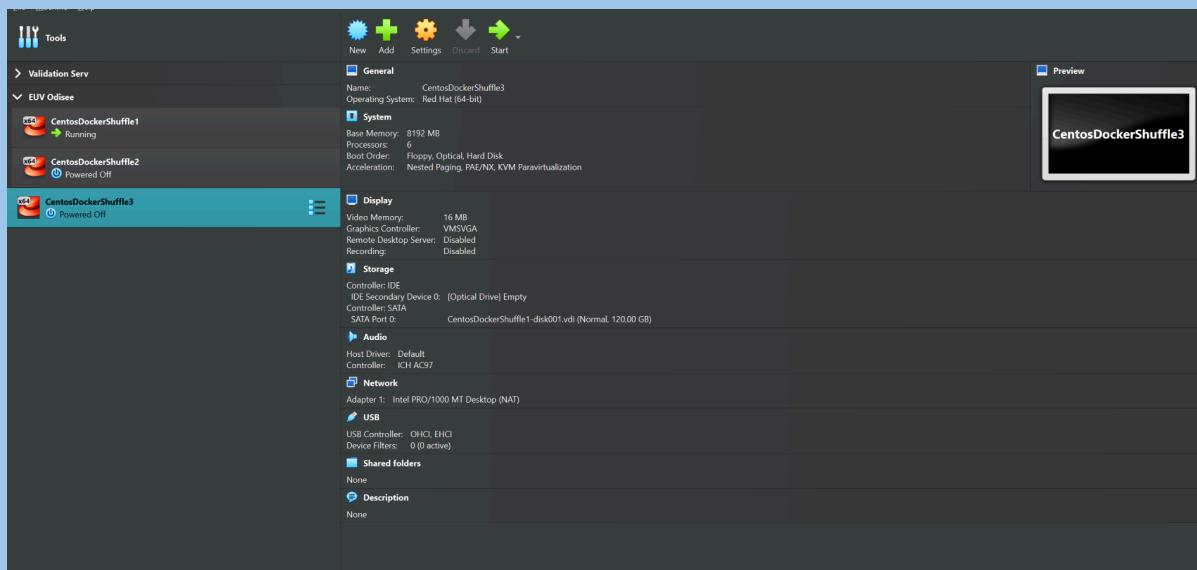
Raadpleeg de online [video](#) om het volledige proces te bekijken.



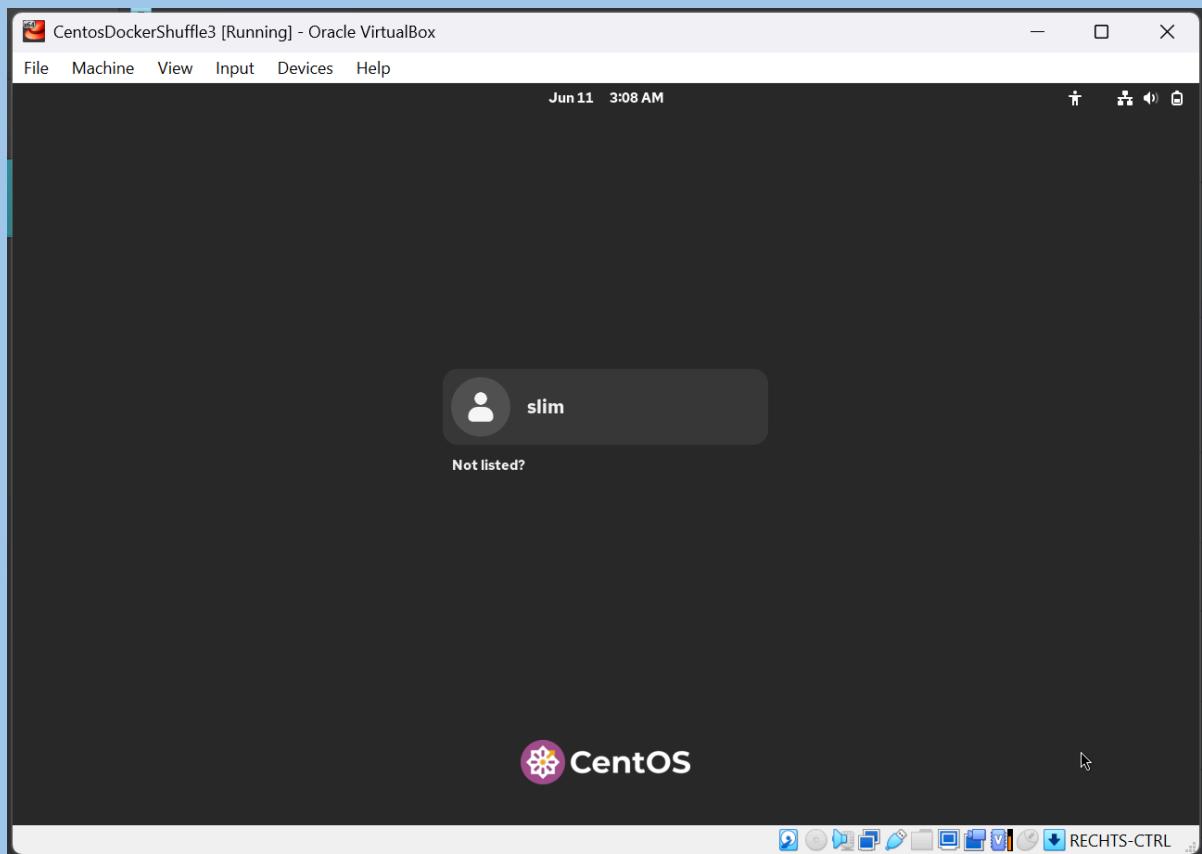
Figuur 6: import van een ova file



Figuur 6.1: appliances import process



Figuur 6.2: shuffle3 en zijn technische specs



Figuur 6.5: inlog scherm CentOS.

## Proof Of Concept

### Handleiding Overzicht

Deze handleiding bevat de procedure om een volledig geautomatiseerd Eurovision stemsysteem op te stellen.

### Project Setup en Configuratie

### Vereisten

- CentOS Stream 10 VM
- Docker & Docker Compose
- Python 3.11+
- setupinfra.py

## Uitvoering

# Stap 1: Maak een folder aan en navigeer binnen deze folder

mkdir ~/scripts

cd ~/scripts

# Stap 2: Voer setup script uit

download the setupinfra.py en plaats hem in je folder voor dat je onderstaande commando uitvoert

python3 setupinfra.py

```
((euv-local) ) [slim@vbox scripts]$ python setupinfra.py
Enter the name of the general project folder (default: Euvsong): demo_install_infrastructure
Should the generator be interactive at runtime? [y/n]: y
Enter Docker image name for generator (default: euvsong-generator:latest):
Image 'euvsong-generator:latest' already exists locally. Enter a new name or press Enter to keep it:
Enter Docker image name for aggregator (default: euvsong-aggregator:latest):
Image 'euvsong-aggregator:latest' already exists locally. Enter a new name or press Enter to keep it:
Enter Docker image name for analyzer (default: euvsong-analyzer:latest):
Image 'euvsong-analyzer:latest' already exists locally. Enter a new name or press Enter to keep it:
Set permissions 777 on demo_install_infrastructure/data
Written demo_install_infrastructure/generator/generate_votes.py
Written demo_install_infrastructure/generator/Dockerfile.generator
Written demo_install_infrastructure/aggregator/aggregate_votes.py
Written demo_install_infrastructure/aggregator/Dockerfile.aggregator
Written demo_install_infrastructure/analyzer/final_ranking.py
Written demo_install_infrastructure/analyzer/Dockerfile.analyzer
Written demo_install_infrastructure/docker-compose.yml

Summary of actions:
demo_install_infrastructure/generator/generate_votes.py: written
demo_install_infrastructure/generator/Dockerfile.generator: written
demo_install_infrastructure/aggregator/aggregate_votes.py: written
demo_install_infrastructure/aggregator/Dockerfile.aggregator: written
demo_install_infrastructure/analyzer/final_ranking.py: written
demo_install_infrastructure/analyzer/Dockerfile.analyzer: written
demo_install_infrastructure/docker-compose.yml: written

Running 'docker compose build' in demo_install_infrastructure ...
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 3.1s (26/26) FINISHED
=> [generator internal] load build definition from Dockerfile.generator          docker:default      0.0s
=> => transferring dockerfile: 231B                                            0.0s
=> [analyzer internal] load metadata for docker.io/library/python:3.11-slim      1.6s
=> [generator internal] load .dockerrignore                                     0.0s
=> => transferring context: 2B                                                 0.0s
=> [analyzer 1/3] FROM docker.io/library/python:3.11-slim@sha256:6169864086d7e3086c236dd7fdbdb127b4e6e90fd3c7825fd9aa0e547578e 0.0s
=> [generator internal] load build context                                     0.0s
=> => transferring context: 4.68kB                                           0.0s
=> CACHED [generator 2/4] RUN pip install pandas numpy                         0.0s
=> CACHED [generator 3/4] WORKDIR /app                                         0.0s
=> CACHED [generator 4/4] COPY generate_votes.py .                           0.0s
```

```

--> CACHED [generator 4/4] copy aggregate_votes.py .
=> [generator] exporting to image
=> => exporting layers
=> => writing image sha256:ae4529134b832b01305cb855148d5a30c4573a392130eb9664b07f86bbef226
=> => naming to docker.io/library/euvsong-generator:latest
=> [generator] resolving provenance for metadata file
=> [aggregator internal] load build definition from Dockerfile.aggregator
=> => transferring dockerfile: 215B
=> [aggregator internal] load metadata for docker.io/bitnami/spark:latest
=> [aggregator internal] load .dockerrcignore
=> => transferring context: 2B
=> [aggregator 1/3] FROM docker.io/bitnami/spark:latest@sha256:8112420cf8c553528e50e6452b9d3a09f4d1d436b7186527070c121b01bd
=> [aggregator internal] load build context
=> => transferring context: 1.36kB
=> CACHED [aggregator 2/3] WORKDIR /app
=> CACHED [aggregator 3/3] COPY aggregate_votes.py .
=> [aggregator] exporting to image
=> => exporting layers
=> => writing image sha256:9c24b890e3a964d92ee895305abe8e4699a648a2b369d9e1f78018e10b0a2fb5
=> => naming to docker.io/library/euvsong-aggregator:latest
=> [aggregator] resolving provenance for metadata file
=> [analyzer internal] load build definition from Dockerfile.analyzer
=> => transferring dockerfile: 199B
=> [analyzer internal] load .dockerrcignore
=> => transferring context: 2B
=> [analyzer internal] load build context
=> => transferring context: 2.35kB
=> CACHED [analyzer 2/3] WORKDIR /app
=> CACHED [analyzer 3/3] COPY final_ranking.py .
=> [analyzer] exporting to image
=> => exporting layers
=> => writing image sha256:5bfb71945d7dd539d9715c3fc606a022301ff3fad9454ad60f097a394e77685a
=> => naming to docker.io/library/euvsong-analyzer:latest
=> [analyzer] resolving provenance for metadata file
[+] Building 3/3
✓ aggregator    Built
✓ analyzer     Built
✓ generator   Built
Docker images built successfully.

To run the full pipeline:
cd demo_install_infrastructure
docker compose run --rm generator
docker compose up aggregator analyzer

```

Figuur 7: installatie van een infrastructuur met behulp van het setupscript

# Stap 3: Configureer project parameters

# - Project naam: demo\_install\_infra

# - Interactieve modus: y

```
# - Docker images: demo-generator:latest, demo-aggregator:latest, demo-analyzer:latest
```

## Resultaat

```
((euv-local) ) [slim@vbox demo_install_infrastructure]$ ls -lra
total 4
drwxrwxrwx. 2 slim slim 6 Jun 11 05:44 data
drwxr-xr-x. 5 slim slim 154 Jun 11 05:44 ..
drwxr-xr-x. 2 slim slim 59 Jun 11 05:44 generator
drwxr-xr-x. 2 slim slim 57 Jun 11 05:44 analyzer
drwxr-xr-x. 2 slim slim 61 Jun 11 05:44 aggregator
-rw-r--r--. 1 slim slim 889 Jun 11 05:44 docker-compose.yml
drwxr-xr-x. 6 slim slim 95 Jun 11 05:44 .
((euv-local) ) [slim@vbox demo_install_infrastructure]$ █
```

Figuur 7.1 Complete projectstructuur met Docker containers en orkestratie files (docker-compose.yml).

## Generator Container - Stemdata

Genereert realistische Eurovision stemdata met authentieke mobiele nummers voor 11 EU landen.

### Technische Specificaties

- Base Image: python:3.11-slim
- Dependencies: pandas, numpy
- Output: CSV bestanden met 100.000 stemmen per land
- Seed: 42 (voor reproduceerbare resultaten)

### Uitvoering

```
cd demo_install_infra
```

```
docker compose run --rm generator
```

```
# Interactieve keuze:
```

```
# "Type 'IT' for Italy only or 'ALL' for all countries: IT"
```

## Data Formaat

COUNTRY CODE,MOBILE NUMBER,SONG NUMBER,TIMESTAMP

IT,+39 345 123 456,4,2025-06-11T04:17:23

IT,+39 387 654 321,19,2025-06-11T03:45:12

## Mobiele Nummer Formaten[4]

- België: +32 4XX XXX XXX
- Frankrijk: +33 6 XX XX XX XX
- Duitsland: +49 15X XXX XXXX
- Italië: +39 3XX XXX XXX
- [8 andere landen met authentieke formaten]

## Aggregator Container - Apache Spark Verwerking

### Functionaliteit

Verwerkt ruwe stemdata met Apache Spark voor gedistribueerde computing en statistische analyse.

### Technische Specificaties

- Base Image: bitnami/spark:latest (Spark 4.0.0)
- Java: OpenJDK 17.0.15
- Processing: Map-Reduce operaties
- Input: Alle \*\_votes.txt bestanden
- Output: italy\_votes\_reduced.json

## Spark Workflow[3]

```
# Map fase: (land, liedje) → 1  
  
mapped = rdd.map(lambda line: line.strip().split(",")) \  
  
.map(lambda fields: ((fields[0], fields[2]), 1))
```

## # Reduce fase: Tel stemmen per combinatie

```
reduced = mapped.reduceByKey(lambda a, b: a + b)
```

## # Group fase: Groepeer per land

```
grouped = reduced.map(lambda x: (x[0][0], (x[0][1], x[1]))) \n\n    .groupByKey() \\n\n    .mapValues(list)
```

## Uitvoering

```
docker compose run --rm aggregator
```

## Performance Metrics

- Total votes loaded: 100.000 (voor IT alleen)
  - Processing tijd: ~6.8 seconden
  - Spark stages: 3 (ShuffleMapStage 0 & 1, ResultStage 2)
  - Memory usage: 434.4 MiB RAM

## Analyzer Container - Finale Ranking

### Functionaliteit

Analyseert geaggregeerde data en genereert professionele Eurovision rankings met mooie banners.

### Technische Specificaties

- Base Image: python:3.11-slim
- Input: italy\_votes\_reduced.json
- Output: Gekleurde console output + italy\_ranking.txt

### Uitvoering

```
docker compose run --rm analyzer
```

### Output Voorbeeld

```
|| 🎉 EUROVISION CHAMPION! 🎉 ||  
||  
|| 🏆 Overall Winner: Song 4 with 4129 votes ||  
||
```

### Eurovision Overall Vote Ranking:

1. Song 4: 4129 votes

2. Song 19: 4115 votes

3. Song 21: 4066 votes

[...]

Docker Compose Orchestrator

## Automatische Pipeline

services:

generator:

image: demo-generator:latest

volumes:

- ./data:/data

stdin\_open: true

tty: true

aggregator:

image: demo-aggregator:latest

depends\_on:

- generator

volumes:

- ./data:/data

healthcheck:

test: ["CMD", "sh", "-c", "test -f /data/italy\_votes\_reduced.json"]

analyzer:

image: demo-analyzer:latest

depends\_on:

aggregator:

```
condition: service_completed_successfully
```

```
volumes:
```

```
- ./data:/data
```

## Volledige Pipeline Uitvoering

```
# Optie 1: Stap voor stap (interactief)
```

```
docker compose run --rm generator
```

```
docker compose run --rm aggregator
```

```
docker compose run --rm analyzer
```

```
# Optie 2: Volledig geautomatiseerd
```

```
docker compose up
```

## Data Flow en Bestandsstructuur

### Project Layout

```
demo_install_infra/
```

```
    |--- data/          # Shared volume
    |
    |   |--- it_votes.txt      # Raw votes (4.0MB)
    |
    |   |--- italy_votes_reduced.json # Aggregated (2.4KB)
    |
    |       |--- italy_ranking.txt    # Final ranking (678B)
    |
    |--- generator/
    |
    |   |--- generate_votes.py
```

```
|   └── Dockerfile.generator  
|  
|   ├── aggregator/  
|   |   ├── aggregate_votes.py  
|   |  
|   └── Dockerfile.aggregator  
|  
|   ├── analyzer/  
|   |   ├── final_ranking.py  
|   |  
|   └── Dockerfile.analyzer  
|  
└── docker-compose.yml
```

## Data Transformatie

Raw CSV → Spark Processing → JSON → Analysis → TXT

100K records → Map-Reduce → 25 songs → Ranking → Winner

## Troubleshooting en Debugging

### Problemen

#### 1. Container Build Fouten

# Oplossing: Rebuild zonder cache

docker compose build --no-cache

#### 2. Permissie Problemen

# Oplossing: Fix data folder permissies

chmod 777 demo\_install\_infra/data

### 3. Spark Memory Issues

```
# Oplossing: Verhoog container memory  
  
docker run --memory="2g" demo-aggregator:latest
```

### 4. Interactieve Mode Problemen

```
# Oplossing: Gebruik environment variables  
  
docker run -e VOTE_COUNTRY_MODE="IT" demo-generator:latest
```

```
# View logs  
  
docker compose logs aggregator
```

```
# Monitor resources  
  
docker stats
```

### Data Validation

```
# Verify output files  
  
ls -la demo_install_infra/data/  
  
wc -l demo_install_infra/data/it_votes.txt # Should be 100000
```

## Gedistribueerd Concept

Deze [video](#) demonstreert stap voor stap de werking van de gedistribueerde versie van mijn Eurovision stem automatiseringssysteem. In het kader van mijn

afstudeerproject heb ik verschillende methodieken geïmplementeerd die tijdens retrospectieve evaluatie sessies zijn geanalyseerd en geoptimaliseerd.

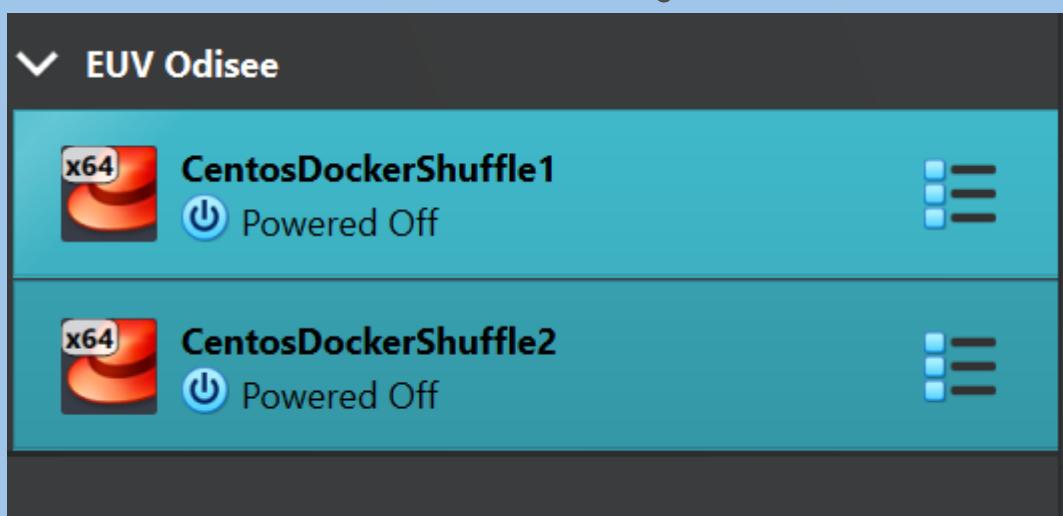
Infrastructuur Architectuur:

Het systeem gebruikt Google Drive als centrale cloud storage oplossing voor zowel data-inname als resultaat-output, wat zorgt voor een schaalbare en betrouwbare data pipeline.

VM configuratie voor ons gedistribueerd versie:

In tegenstelling tot het Proof Of Concept draait onze docker container elke op een afzonderlijke server om een echte situatie te simuleren.

Hiervoor hebben we twee CentOS servers aangemaakt.



Figuur 8: VM shuffle 1 en shuffle 2 simulatie gedistribueerd infra

CentosDockerShuffle1 heeft als doel het inlezen van de stemmingen beschikbaar in de data folder en deze te consolideren in een JSON formaat in de output folder op onze google drive storage.

Via SSH of met een rechtstreekse verbinding op de server moet je de volgende commando uitvoeren om Shuffle1 te initieren.

```
docker run \
--name eurovision-aggregator-$(date +%Y%m%d-%H%M%S) \
--rm \
-v ~/eurovision-poc/credentials.json:/app/credentials.json:ro \
-e DATA_FOLDER_ID="13k5JQf1rPm3ypdQcp_xKD_BmrlwJb8RR" \
-e OUTPUT_FOLDER_ID="1nSisXt0twfua8OaJbKax1ITsDLUSRZN5" \
--memory="4g" \
--cpus="2.0" \
```

```
shuffle1:latest
```

Dit commando start een tijdelijke container gebaseerd op de `shuffle1` Docker image, met gecontroleerde toegang tot resources (CPU & RAM), geef je hem een dynamische naam, monteer je een credentials-bestand, en geef je hem twee map-ID's mee als omgevingsvariabelen. Dit alles wordt automatisch opgeschoond zodra het proces klaar is.

`CentosDockerShuffle2` verwerkt het JSON file beschikbaar in output bevestig het winaar en bezorg de resultaten zowel in json formaat als html.

```
docker run \
--name eurovision-analyzer-$(date +%Y%m%d-%H%M%S) \
--rm \
-v ~/eurovision-poc/credentials.json:/app/credentials.json:ro \
-e OUTPUT_FOLDER_ID="1nSisXt0twfua8OaJbKax1ITsDLUSRZN5" \
shuffle2:latest
```

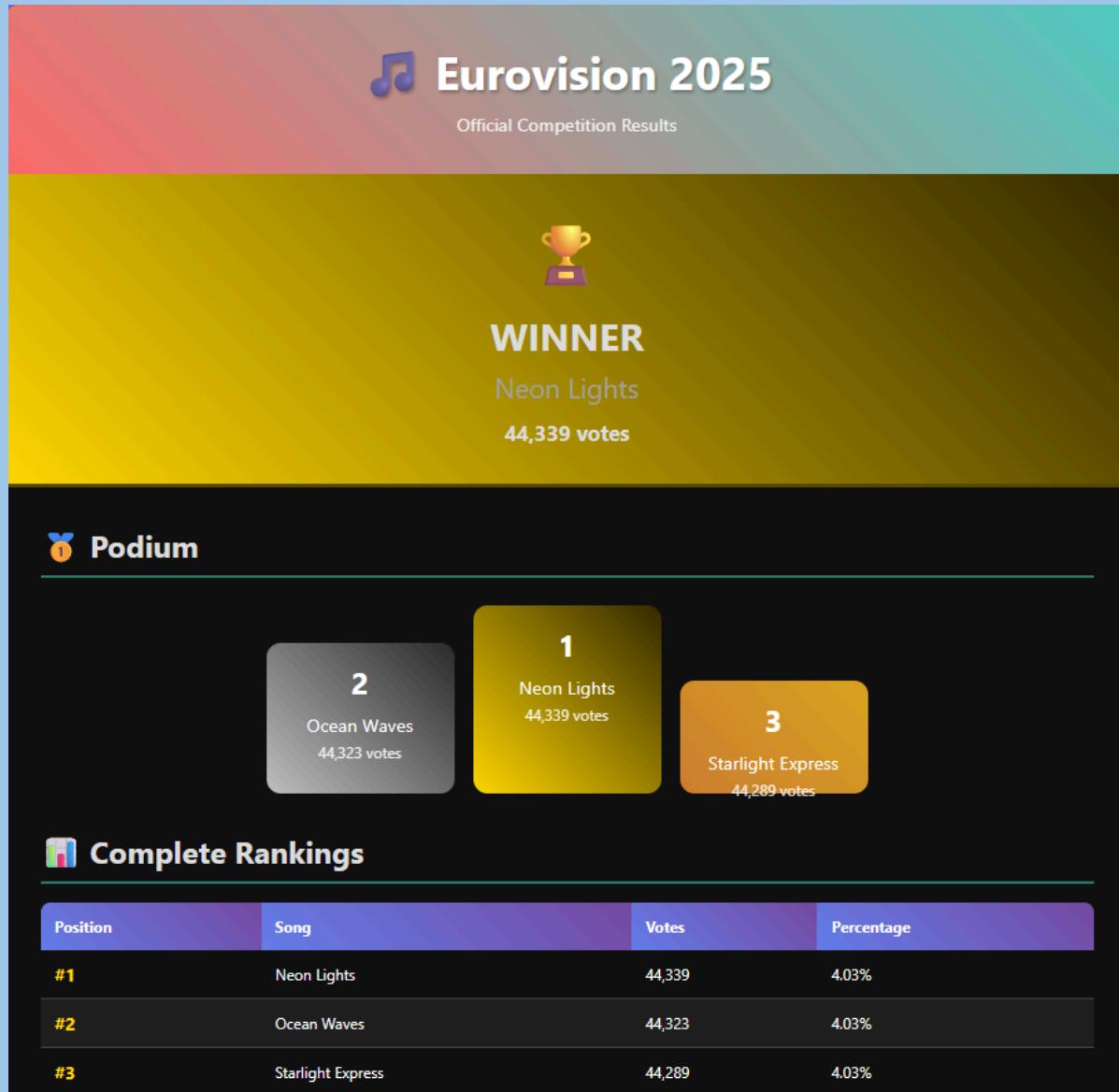
Dit commando start een tijdelijke Docker-container op basis van de image `shuffle2:latest`, met als doel het analyseren van Eurovision-data. De container:

- Krijgt een unieke naam met datum en tijd (bv. `eurovision-analyzer-20250611-154200`).
- Verwijdert zichzelf automatisch na afloop (`--rm`).
- Krijgt toegang tot een read-only Google credentials-bestand van de host.
- Krijgt via een omgevingsvariabele (`OUTPUT_FOLDER_ID`) de ID van een Google Drive-map waarin de analyse-resultaten worden opgeslagen.

**Shuffle2** zorgt dat de winnaar weergetoont wordt.

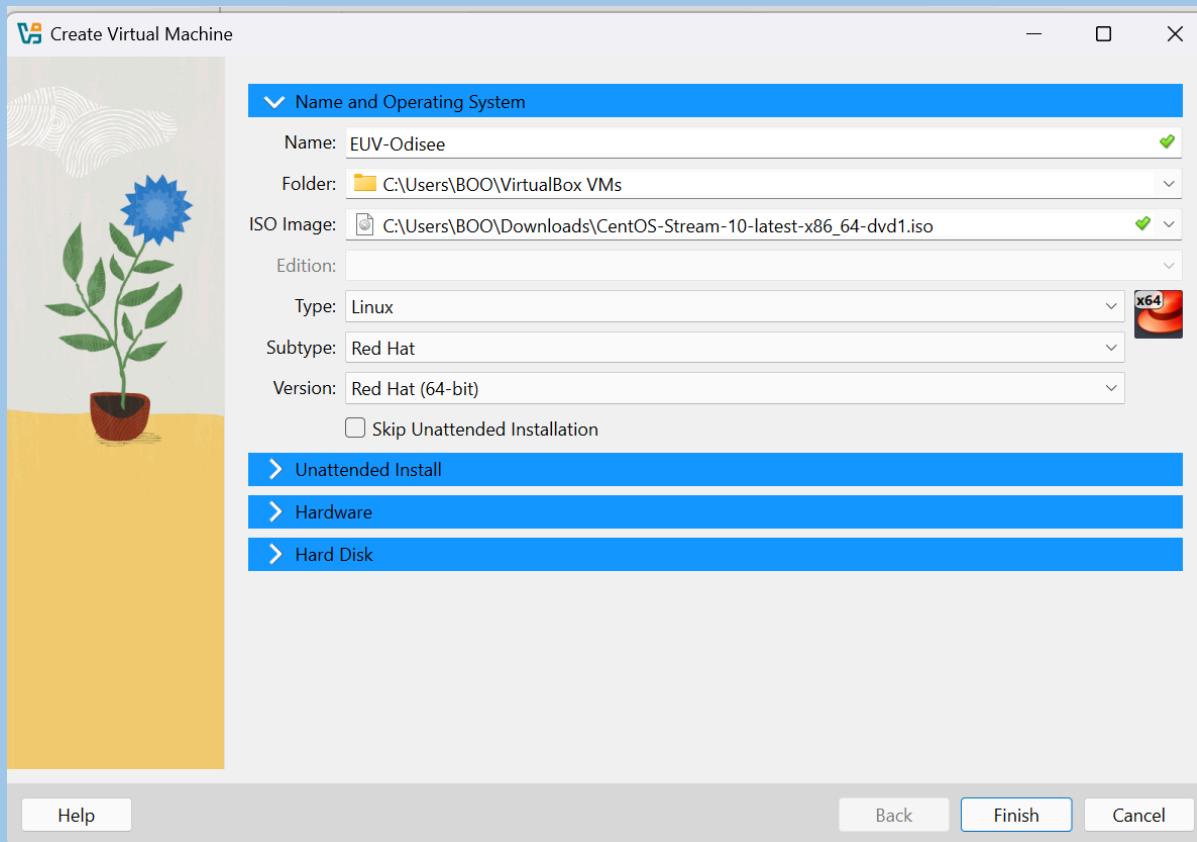
```
{  
  "summary": {  
    "total_votes_processed": 1100000,  
    "total_songs": 25,  
    "total_countries": 11  
  },  
  "song_rankings": [  
    {  
      "song_number": 9,  
      "total_votes": 44339  
    },  
    {  
      "song_number": 4,  
      "total_votes": 44323  
    },  
    {  
      "song_number": 21,  
      "total_votes": 44289  
    },  
    {  
      "song_number": 22,  
      "total_votes": 44222  
    },  
    {  
      "song_number": 2,  
      "total_votes": 44183  
    },  
    {  
      "song_number": 18,  
      "total_votes": 44157  
    },  
    {  
      "song_number": 13,  
      "total_votes": 44153  
    },  
    {  
      "song_number": 1,  
      "total_votes": 44121  
    },  
    {  
      "song_number": 11,  
      "total_votes": 44075  
    },  
    {  
      "song_number": 10,  
      "total_votes": 44075  
    }  
  ]  
}
```

Figuur 8.1: Verwerkte data in het JSON bestand

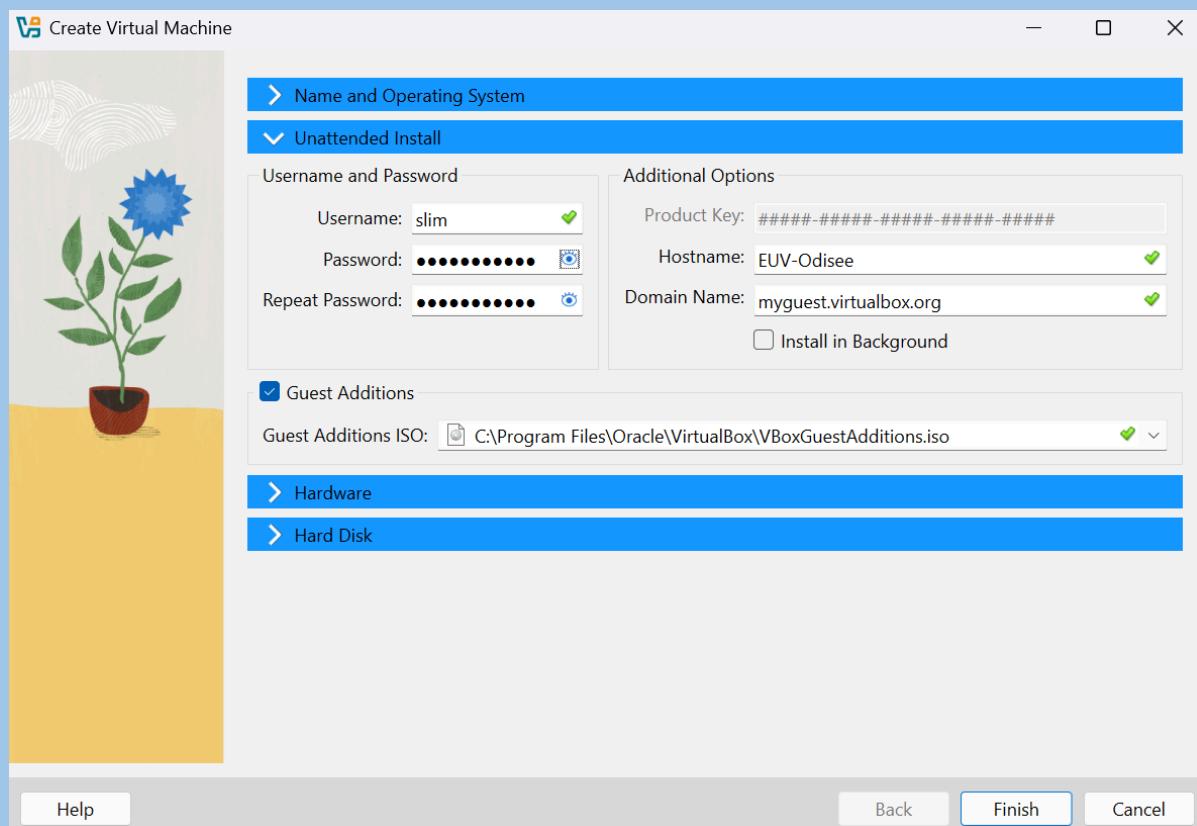


Figuur 8.2 geformateerd resultaten in een mooie layout voor webbrowser .

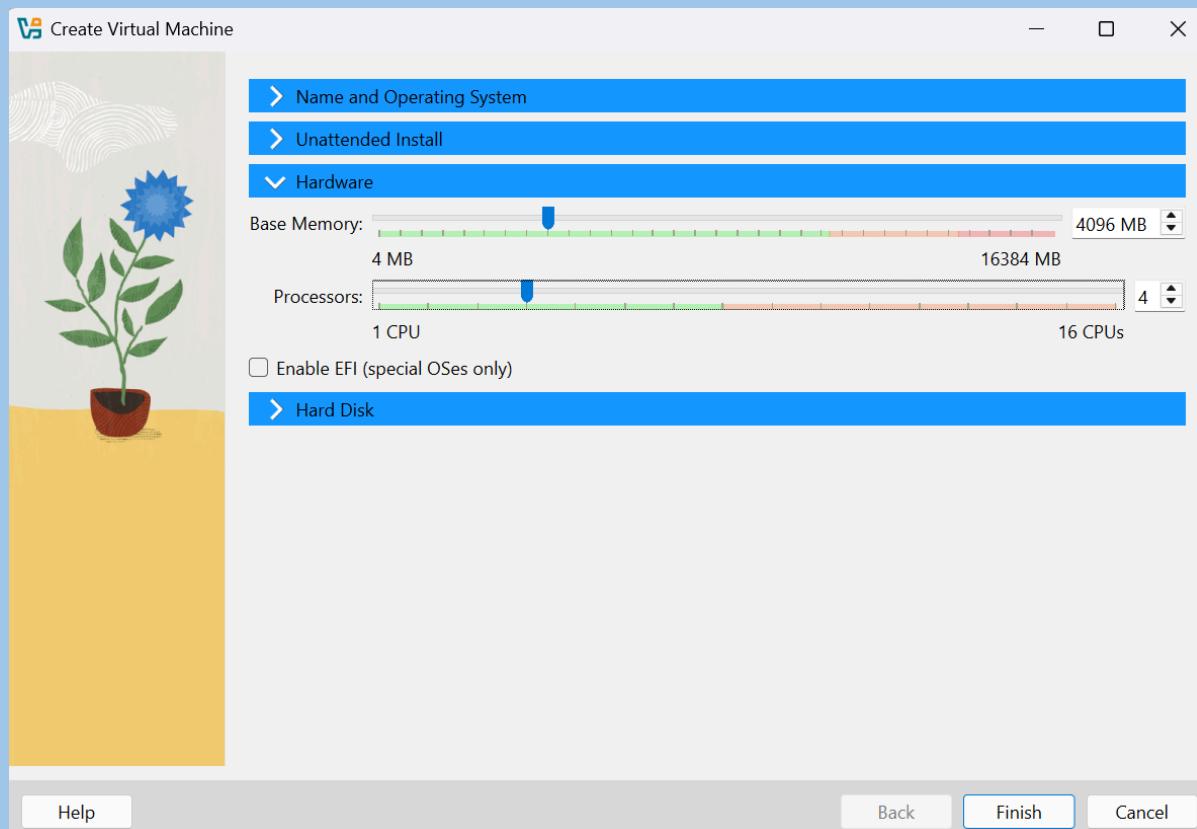
## Full Installation + Rclone



Figuur 9: Installatie van een nieuwe CentOS VM met een ISO



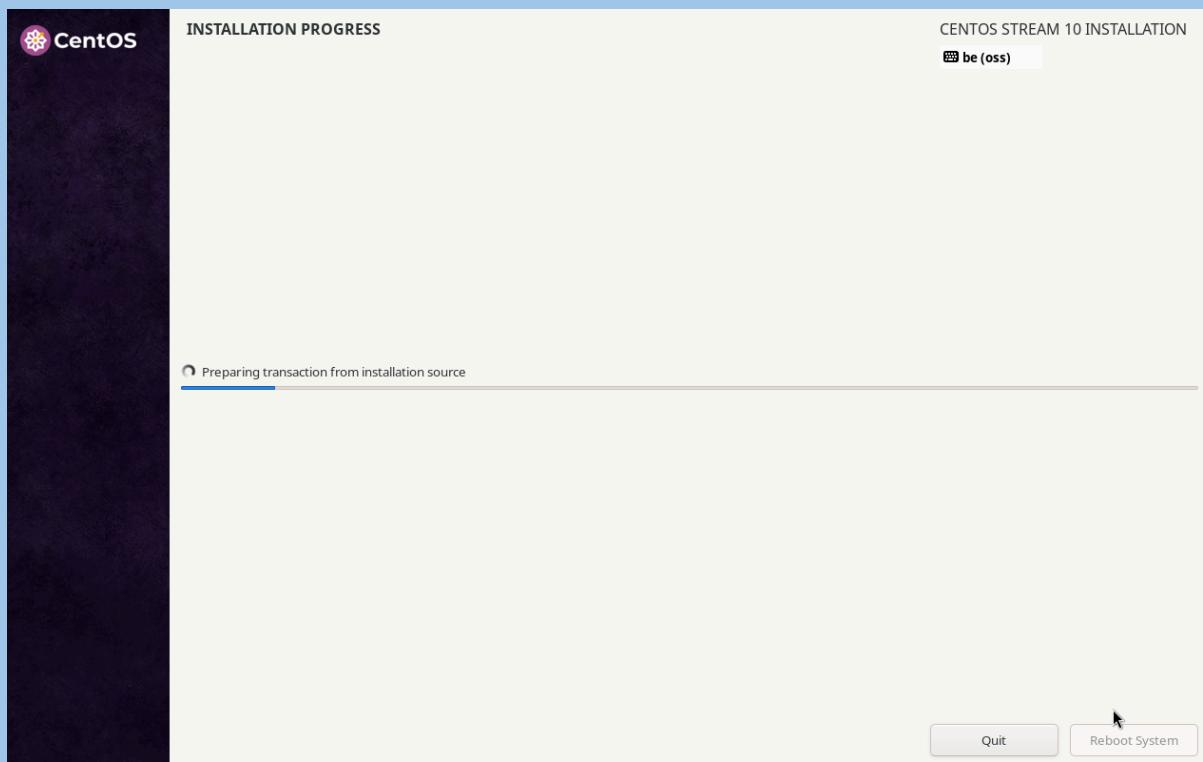
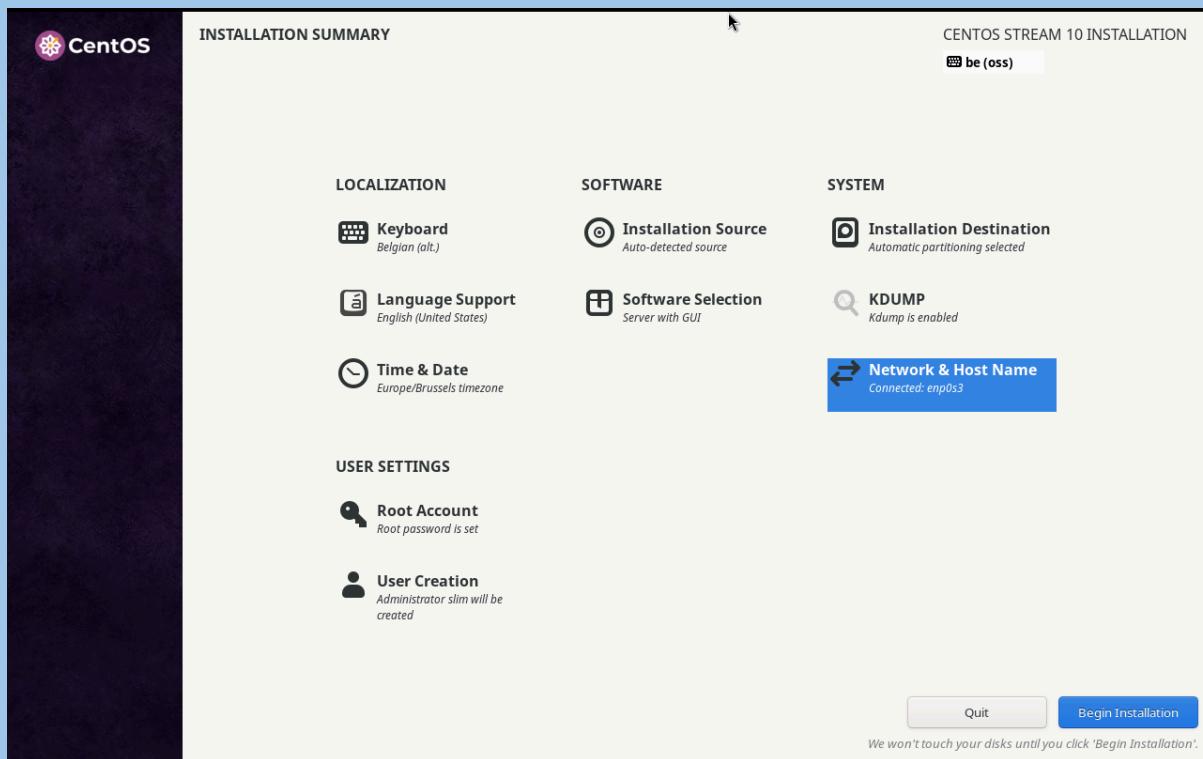
Figuur 9.1 instellen van lokaal account

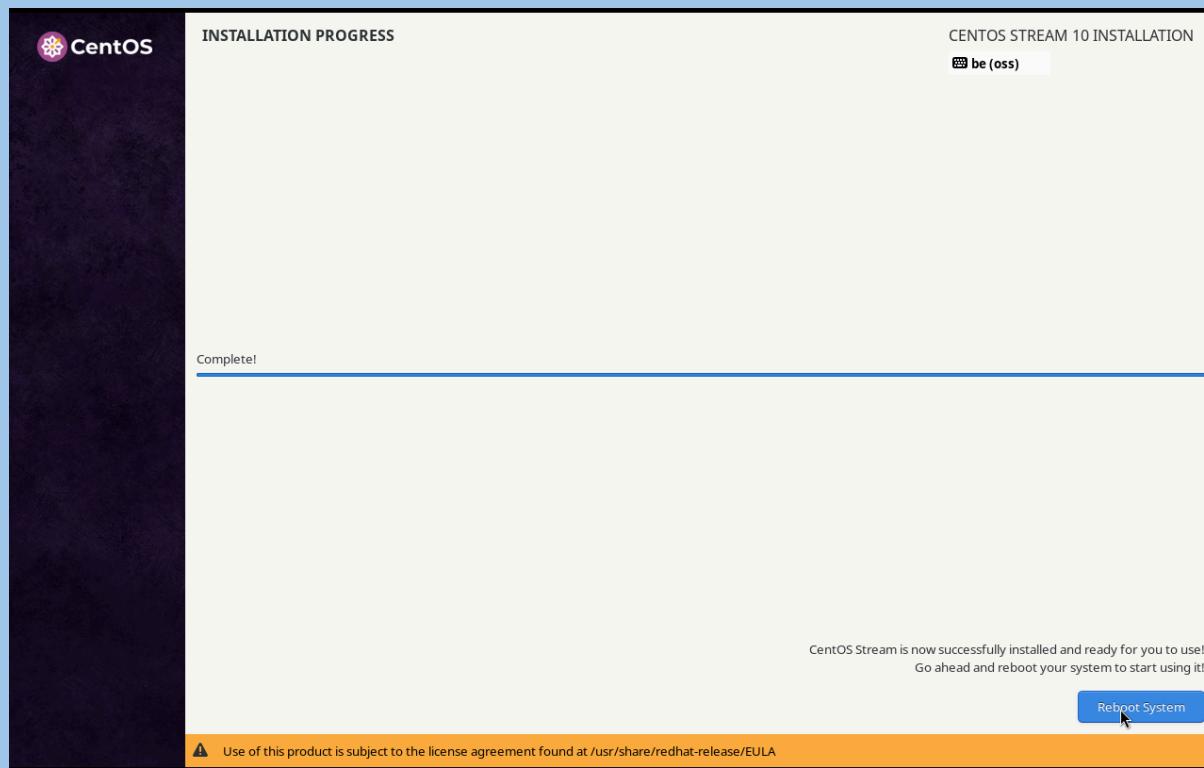


figuur 9.2 configuratie van minimaal RAM en CPU

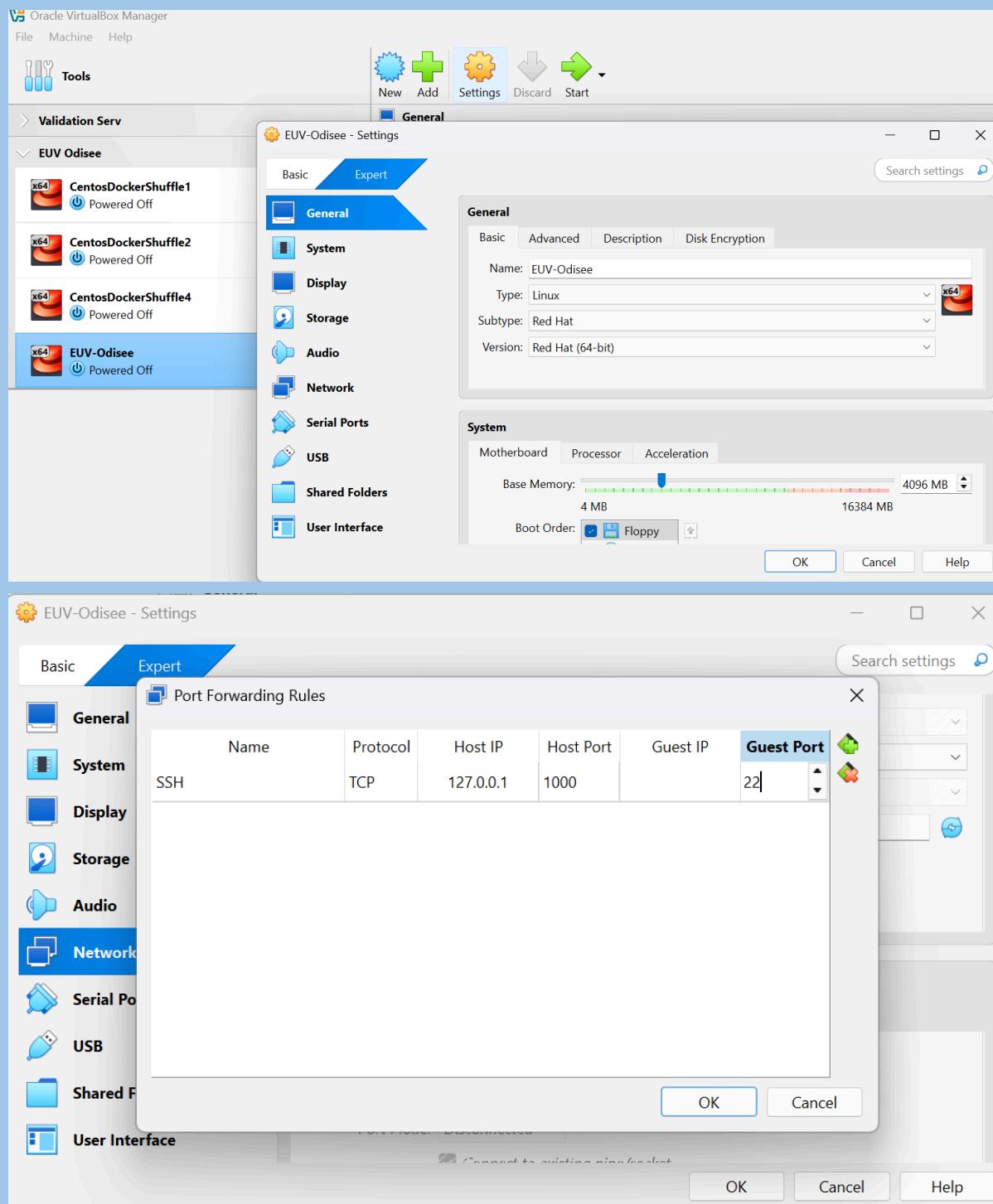


Figuur 9.3 installatie scherm van CentOS bij eerste boot

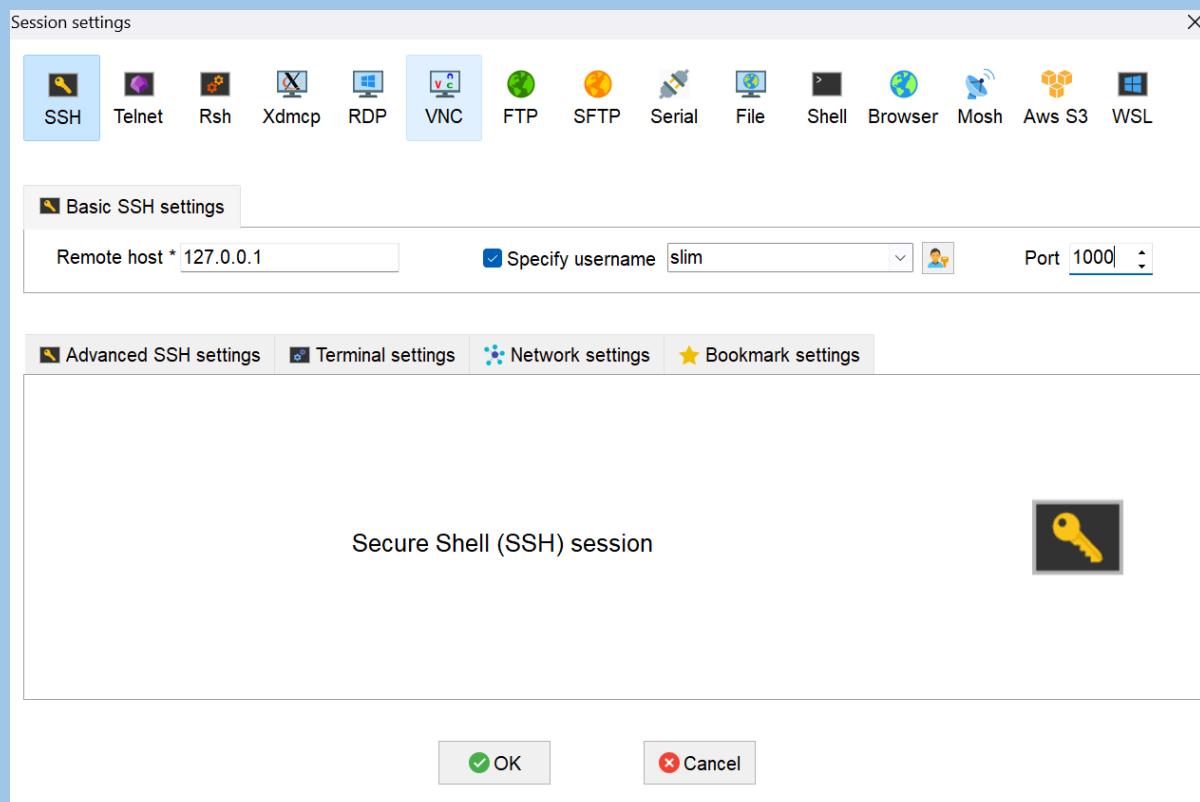




Figuur 9.4 installatie process



figuur 9.5 VM - SSH Port Forwarding Rules



figuur 9.6 mobaxterm login screen - ssh connectie

```
[slim@vbox ~]$ more /etc/os-release
NAME="CentOS Stream"
VERSION="10 (Coughlan)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="10"
PLATFORM_ID="platform:el10"
PRETTY_NAME="CentOS Stream 10 (Coughlan)"
ANSI_COLOR="0;31"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:centos:centos:10"
HOME_URL="https://centos.org/"
VENDOR_NAME="CentOS"
VENDOR_URL="https://centos.org/"
BUG_REPORT_URL="https://issues.redhat.com/"
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux 10"
REDHAT_SUPPORT_PRODUCT_VERSION="CentOS Stream"
[slim@vbox ~]$
```

figuur 9.7 release versie van lokaal installatie

```

generate_votes.py
[slim@vbox generator]$ python3 generate_votes.py
Traceback (most recent call last):
  File "/home/slimg/EUV-Odisee/generator/generate_votes.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
[slim@vbox generator]$ cat > Dockerfile << EOF
FROM python:3.11-slim

WORKDIR /app

# Installeer dependencies
RUN pip install pandas numpy

# Kopieer script
COPY generate_votes.py .

# Maak data directory
RUN mkdir -p /data

CMD ["python", "generate_votes.py"]
EOF
[slim@vbox generator]$ ls
Dockerfile generate_votes.py
[slim@vbox generator]$ sudo docker build -t generator:latest -f Dockerfile .
[sudo] password for slim:
[+] Building 2.3s (10/10) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 312B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim          1.6s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                             0.0s
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:9e1912aab0a30bb9488eb79063f68f42a68ab0946cbe98fecf197fe5b0855  0.0s
=> [internal] load build context                                              0.0s
=> => transferring context: 2.95kB                                         0.0s
=> CACHED [2/5] WORKDIR /app                                                 0.0s
=> CACHED [3/5] RUN pip install pandas numpy                                0.0s
=> [4/5] COPY generate_votes.py .                                           0.0s
=> [5/5] RUN mkdir -p /data                                                 0.3s
=> exporting to image                                                       0.1s
=> => exporting layers                                                       0.1s
=> => writing image sha256:e64745a0c15bc9839c00d7a0acf6312f5ee3bdcedf7b7bda21549c915c1688a1  0.0s
=> => naming to docker.io/library/generator:latest                           0.0s
[slim@vbox generator]$ sudo docker run --rm -it \
-v ~/EUV-Odisee/data:/data \
generator:latest
Do you want to generate votes for Italy only or all EU festival countries?
Type 'IT' for Italy only or 'ALL' for all countries: IT
File '/data/it_votes.txt' has been saved.
[slim@vbox generator]$ 

```

figuur 9.8 generatie van fictief votes om stemmen te simuleren

`sudo docker run --rm -it \`

`-v ~/EUV-Odisee/data:/data \`

`generator:latest`

```

[slim@vbox generator]$ docker run --rm \
-v ~/EUV-Odisee/data:/data:ro \
-v ~/EUV-Odisee/output:/output \
shuffle1:latest
spark 17:08:58.66 INFO  ===>
spark 17:08:58.67 INFO  ===> Welcome to the Bitnami spark container
spark 17:08:58.67 INFO  ===> Subscribe to project updates by watching https://github.com/bitnami/containers
spark 17:08:58.67 INFO  ===> Did you know there are enterprise versions of the Bitnami catalog? For enhanced secure software supply chain features, unli
, see Bitnami Premium or Tanzu Application Catalog. See https://www.arrow.com/globalecs-na/vendors/bitnami/ for more information.
spark 17:08:58.67 INFO  ===>

WARNING: Using incubator modules: jdk.incubator.vector
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/06/11 17:09:05 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Processing /data/it_votes.txt...
Results saved to /output/italy_votes_reduced.json
[slim@vbox generator]$ 

```

figuur 9.9 draaiend shuffle1 met lokaal toegang tot folders

`docker run --rm \`

`-v ~/EUV-Odisee/data:/data:ro \`

`-v ~/EUV-Odisee/output:/output \`

`shuffle1:latest`

```
[slim@vbox generator]$ docker run --rm \
-v ~/EUV-Odisee/output:/output:ro \
-v ~/EUV-Odisee/results:/results \
shuffle2:latest
```

```
♪ EUROVISION CHAMPION! ♪  
✿ Overall Winner: Song 4 with 4129 votes
```

#### Eurovision Overall Vote Ranking:

1. Song 4: 4129 votes
2. Song 19: 4115 votes
3. Song 21: 4066 votes
4. Song 25: 4065 votes
5. Song 1: 4059 votes
6. Song 9: 4050 votes
7. Song 14: 4048 votes
8. Song 16: 4038 votes
9. Song 20: 4010 votes
10. Song 17: 4004 votes
11. Song 2: 4004 votes
12. Song 23: 4003 votes
13. Song 18: 3999 votes
14. Song 10: 3995 votes
15. Song 5: 3992 votes
16. Song 13: 3992 votes
17. Song 6: 3991 votes
18. Song 22: 3982 votes
19. Song 15: 3961 votes
20. Song 24: 3952 votes
21. Song 8: 3952 votes
22. Song 7: 3945 votes
23. Song 12: 3933 votes
24. Song 3: 3917 votes
25. Song 11: 3798 votes

Overall Winner: Song 4 with 4129 votes

```
Results saved to /results/italy_ranking.txt
HTML results saved to /results/eurovision_results.html
[slim@vbox generator]$
```

Figuur 9.10 draaiend shuffle2 met lokaal toegang tot folders

```
docker run --rm \
-v ~/EUV-Odisee/output:/output:ro \
-v ~/EUV-Odisee/results:/results \
shuffle2:latest
```

# Eurovision 2025 Official Results

 **Winner: Song 4 with 4129 votes**

## Final Ranking:

1. Song 4: 4129 votes
2. Song 19: 4115 votes
3. Song 21: 4066 votes
4. Song 25: 4065 votes
5. Song 1: 4059 votes
6. Song 9: 4050 votes
7. Song 14: 4048 votes
8. Song 16: 4038 votes
9. Song 20: 4010 votes
10. Song 17: 4004 votes
11. Song 2: 4004 votes
12. Song 23: 4003 votes
13. Song 18: 3999 votes
14. Song 10: 3995 votes
15. Song 5: 3992 votes
16. Song 13: 3992 votes
17. Song 6: 3991 votes
18. Song 22: 3982 votes
19. Song 15: 3961 votes
20. Song 24: 3952 votes
21. Song 8: 3952 votes
22. Song 7: 3945 votes
23. Song 12: 3933 votes
24. Song 3: 3917 votes
25. Song 11: 3798 votes

Figuur 10 Eindresultaat HTML pagina.

## GITHUB

### Git-initialisatie met een bestaande GitHub-repository

#### 1. Gebruikersconfiguratie instellen (éénmalig):

Voer deze commando's uit om je naam en e-mailadres op te geven:

```
git config --global user.name "Jouw Naam"
```

```
git config --global user.email "jouw.email@example.com"
```

```
[slim@vbox results]$ git version
git version 2.47.1
[slim@vbox results]$ git config --global user.name "slimane"
git config --global user.email "slimane.elboujadaini@student.odisee.be"
[slim@vbox results]$ git config --list
user.name=slimane
user.email=slimane.elboujadaini@student.odisee.be
```

figuur 10 git configuratie

## 2. Ga naar je projectmap:

```
cd /home/slim/EUV-Odisee
```

## 3. Initialiseer Git in de map:

```
git init
```

```
[slim@vbox EUV-Odisee]$ git init
hint: Using 'master' as the name for the initial branch. This default branch na
me
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/slim/EUV-Odisee/.git/
```

figuur 10.1 git init

## 4. Voeg de externe GitHub-repository toe:

```
git remote add origin https://github.com/username/naam-van-repo.git
```

## 5. Als jouw project enkel lokaal bestaat (en de GitHub-repo leeg is):

```
git add .
```

```
git commit -m "Initial commit"
```

```
git push -u origin main
```

om onderstaande probleem om te lossen hebben we via de webconsole de sleutels moeten valideren en een token moeten aanmaken om de synchronisatie toe te laten.

```
[slim@vbox EUV-Odisee]$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/slimanetheone/EUV-Odisee
.git'
[slim@vbox EUV-Odisee]$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/slimanetheone/EUV-Odisee
.git'
[slim@vbox EUV-Odisee]$ ^C
[slim@vbox EUV-Odisee]$ git push -u origin master
Username for 'https://github.com': slimanetheone
Password for 'https://slimanetheone@github.com':
remote: Permission to slimanetheone/EUV-Odisee.git denied to slimanetheone.
fatal: unable to access 'https://github.com/slimanetheone/EUV-Odisee.git/': The requested URL returned error: 403
[slim@vbox EUV-Odisee]$ git push -u origin master
Username for 'https://github.com': slimanetheone
Password for 'https://slimanetheone@github.com':
remote: Permission to slimanetheone/EUV-Odisee.git denied to slimanetheone.
fatal: unable to access 'https://github.com/slimanetheone/EUV-Odisee.git/': The requested URL returned error: 403
[slim@vbox EUV-Odisee]$ git push -u origin master
Username for 'https://github.com': slimanetheone
Password for 'https://slimanetheone@github.com':
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 4 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (22/22), 973.18 KiB | 6.53 MiB/s, done.
Total 22 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote: error: GH013: Repository rule violations found for refs/heads/master.
remote:
```

git error

## 6. Als de GitHub-repository al bestanden bevat:

```
git pull origin main --allow-unrelated-histories
```

```
git push -u origin main
```

```
[slim@vbox EUV-Odisee]$ git push -u origin master
Username for 'https://github.com': slimanetheone
Password for 'https://slimanetheone@github.com':
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 4 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (22/22), 973.18 KiB | 7.54 MiB/s, done.
Total 22 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/slimanetheone/EUV-Odisee.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
[slim@vbox EUV-Odisee]$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
[slim@vbox EUV-Odisee]$ ^C
[slim@vbox EUV-Odisee]$ ^C
[slim@vbox EUV-Odisee]$ █
```

figuur 11 git push origin master resultaat