Elbrus Analytics - Bereitstellungshandbuch

Tobias Schmidt

July 24, 2022

1 Server Infrastruktur

1.1 Initiale Server Konfiguration

Listing 1: Setzen der Zeitzone auf ’Europa/Wien’.

```
elbrus@server:~$ sudo timedatectl set-timezone Europe/Vienna
```

Listing 2: Installieren von dem ’firewalld’ Service.

```
elbrus@server:~$ sudo dnf install firewalld
```

1.2 Git

Listing 3: Installieren von dem VCS ’git’.

```
elbrus@server:~$ sudo yum install -y git
```

1.3 Ablagestruktur

Listing 4: Anlegen der Verzeichnissstruktur.

```
elbrus@server:~$ mkdir /var/elbrus
```

1.4 Node.Js

Listing 5: Installieren des Framworks ’Node.Js’.

```
elbrus@server:~$ sudo dnf -y module install nodejs:12
```

1.5 Python

1.5.1 1 - Automatische Installation

Listing 6: Kopieren des Github Repositorys 'report-generator'.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/report-generator.git
```

Listing 7: Ausführen des 'pythonSourceInstall.sh' Scripts.

```
elbrus@server:~$ bash report-generator/pythonSourceInstall.sh
```

1.5.2 2 - Manuel Installation

Listing 8: Installieren von benötigten Packeten und Abhängigkeiten.

```
elbrus@server:~$ sudo dnf install gcc openssl-devel bzip2-devel\
libffi-devel zlib-devel wget make -y
```

Listing 9: Extrahieren der installierent Dateien.

```
elbrus@server:~$ tar -xf Python-3.10.2.tar.xz
```

Listing 10: Wechseln zu source Verzeichniss. Und ausführen des Konfigurations Scripts.

```
elbrus@server:~$ cd Python-3.10.0 && ./configure --enable-optimizations
```

Listing 11: Starten des build Prozesses.

```
elbrus@server:~Python-3.10.0$ cd make -j $(nproc)
```

Listing 12: Installieren von Python.

```
elbrus@server:~Python-3.10.0$ sudo make install
```

1.5.3 Upgrade von ’pip’

Listing 13: Upgraden von ’pip’.

```
elbrus@server:~$ /usr/local/bin/python3.10 -m pip install --upgrade pip
```

1.6 Rust

Listing 14: Installieren von GNU Compiler Collection.

```
elbrus@server:~$ sudo dnf install gcc -y
```

Listing 15: Installieren von Rust

```
elbrus@server:~$ curl --proto ’=https’ --tlsv1.2 -sSf\
https://sh.rustup.rs/ | sh

...

default host triple: x86_64-unknown-linux-gnu
default toolchain: stable (default)
profile: default
modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

...

stable-x86_64-unknown-linux-gnu installed - rustc 1.62.1 (e092d0b6b 2022-07-16)


Rust is installed now. Great!

To get started you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo’s bin directory ($HOME/.cargo/bin).

To configure your current shell, run:
source "$HOME/.cargo/env"
elbrus@server:~$
```

Listing 16: Laden der Variablen aus dem Terminal Profil.

```
elbrus@server:~$ source ~/.profile
```

Listing 17: Hinzufügen des Befehls Cargo zu dem Pfad.

```
elbrus@server:~$ source ~/.cargo/env
```

1.7 SSH-Keys

Weil der pcap-importer und der report-generator auf zwei verschiedenen Server liegen könnten, muss für die Kommunikation zwischen jenen Sever SSH-Funktionieren.

Dieser Schritt kann übersprungen werden wenn alles auf einem Server installiert wird.

1.7.1 Capture-Server

Listing 18: Anlegen der SSH-Keys

```
elbrus@server:~$ mkdir -p /var/elbrus/shared/.ssh/
elbrus@server:~$ ssh-keygen -t ecdsa -b 256 -f\
/var/elbrus/shared/.ssh/id_report_generator_connection -N ''
```

Listing 19: Übertragen der SSH-Keys auf den Database-Server.

```
elbrus@server:~$ ssh-copy-id -i\
/var/elbrus/shared/.ssh/id_report_generator_connection.pub\
elbrus@10.0.76.220
```

1.7.2 Database-Server

Listing 20: Anlegen der SSH-Keys

```
elbrus@server:~$ mkdir -p /var/elbrus/shared/.ssh/
elbrus@server:~$ ssh-keygen -t ecdsa -b 256 -f\
/var/elbrus/shared/.ssh/id_capture_connection -N ''
```

Listing 21: Übertragen der SSH-Keys auf den Capture-Server.

```
elbrus@server:~$ ssh-copy-id -i\
/var/elbrus/shared/.ssh/id_capture_connection.pub\
elbrus@10.0.76.217
```

2 Datenbank

2.1 Voraussetzungen

Listing 22: Hinzufügen des PostgreSQL Drittanbieter-Repository, um die neuesten PostgreSQL-Pakete zu erhalten.

```
elbrus@server:~$ sudo yum install\
https://download.postgresql.org/pub/repos/yum/reporpms/\
EL-$(rpm -E %{rhel})-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

Listing 23: Erstellen und berarbeiten des Timescale repository.

```
elbrus@server:~$ sudo tee /etc/yum.repos.d/\
timescale_timescaledb.repo <<EOL
[timescale_timescaledb]
name=timescale_timescaledb
baseurl=https://packagecloud.io/timescale/timescaledb\
/el/$(rpm -E %{rhel})/\$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/timescale/timescaledb/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOL
```

Listing 24: Updaten der lokalen Package-Liste.

```
elbrus@server:~$ sudo yum update
```

Listing 25: Installieren von TimescaleDB

```
elbrus@server:~$ sudo dnf -qy module disable postgresql
elbrus@server:~$ sudo dnf install postgresql14 postgresql14-server -y
elbrus@server:~$ sudo dnf install timescaledb-2-postgresql-14 -y
```

2.2 Umgebung Konfigurieren

Listing 26: Initialisieren der Datenbank.

```
elbrus@server:~$ /usr/pgsql-14/bin/postgresql-14-setup initdb
```

Listing 27: Verknüpfen von ’postgresql’ Serive Start mit Serverstart sowie den Service starten.

```
elbrus@server:~$ sudo systemctl enable postgresql-14
elbrus@server:~$ sudo systemctl start postgresql-14
```

Listing 28: var/lib/pgsql/14/data/postgresql.conf - Ändern der folgenden Zeilen

```
- #shared_preload_libraries = ’’
+ shared_preload_libraries = ’timescaledb’

- #listen_addresses = ’localhost’
+ listen_addresses = ’*’
```

Listing 29: var/lib/pgsql/14/data/postgresql.conf - Ändern der folgenden Zeilen

```
# TYPE       DATABASE       USER       ADDRESS       METHOD
+ host       elbrus         elbrus     0.0.0.0/0      trust
```

Listing 30: Anpassen der Datenbank Einstellungen auf die Server Hardware.

```
elbrus@server:~$ sudo timescaledb-tune --pg-config=/usr/\
pgsql-14/bin/pg_config --yes
```

Listing 31: Neustarten des Services um Änderungen zu übernehmen.

```
elbrus@server:~$ sudo systemctl restart postgresql-14
```

2.3 Erstellen der Elbrus-Datenbank

Listing 32: Verbinden mit dem interaktiven Terminal von ’postgres’.

```
elbrus@server:~$ sudo su postgres -c psql
```

Im folgenden Text sind markierte Abschnitte Variablen, welche im darunterliegen SQL geändert werden können, was aus Sichertsgründen dringend empfohlen wird.

1. Die Datenbank elbrus anlegen
2. Die Zeitzone auf Europe/Vienna setzen
3. Den User elbrus mit dem Passwort elbrus123! anlegen
4. Dem User alle rechte auf die voher erstellte Datenbank geben

Listing 33: Auführen von SQL Befehlen.

```
CREATE DATABASE elbrus;
ALTER DATABASE elbrus SET timezone TO 'Europe/Vienna';
CREATE USER elbrus PASSWORD 'elbrus123!';
GRANT ALL ON DATABASE elbrus TO elbrus;
```

Listing 34: Wechseln zu erstellter Datenbank

```
\c elbrus
```

Listing 35: Hinzufügen der TimescaleDB Erweiterung.

```
CREATE EXTENSION IF NOT EXISTS timescaledb;
exit
```

2.4 Installation

Listing 36: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/database.git
```

Listing 37: Anlegen der benötigten Tabellen duch das ausführen von 'init.sql'.

```
elbrus@server:~$ psql -U elbrus -d elbrus -f database/sql/init.sql
```

3 Aufzeichnen der Daten

3.1 Voraussetzungen

Listing 38: Installieren von ’tcpdump’ für das aufzeichnen von Daten.

```
elbrus@server:~$ sudo dnf install tcpdump
```

Listing 39: Anlegen eines Users der Berechtigungen zum ausführen von ’tcpdump’ erhält.

```
elbrus@server:~$ sudo useradd aragog
```

Listing 40: Zuweisen von ’tcpdump’ zu der Gruppe ’aragog’.

```
elbrus@server:~$ sudo chgrp aragog /usr/sbin/tcpdump
```

Listing 41: Ändern der Berechtigungen auf ’tcpdump’.

```
elbrus@server:~$ chmod 750 /usr/sbin/tcpdump
elbrus@server:~$ sudo setcap cap_net_raw,cap_net_admin=eip\
/usr/sbin/tcpdump
```

3.2 Installation

Listing 42: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/capture-device.git
elbrus@server:~/var/elbrus$ mkdir capture
elbrus@server:~/var/elbrus$ cp capture-device/src/* capture
elbrus@server:~/var/elbrus$ rm -rfd capture-device
```

3.3 Umgebung Konfigurieren

3.3.1 1 - Mit Setup Script

Listing 43: Ausführen des setup Scripts

```
elbrus@server:~/var/elbrus$ bash capture/init.sh
Do you want to proceed with setup the ’capture-device’? (y/n) y

Where should the log be stored (dir) [/var/elbrus/shared/log]:
Where is the elb-capture-postrotate.sh stored [/var/elbrus/capture/elb-
 capture-postrotate.sh]:
Where is the shared config stored [/var/elbrus/shared/.config]:

Should the log be stored at ’/var/elbrus/shared/log’ ?
Is the ’elb-capture-postrotate.sh’ stored at ’/var/elbrus/capture/elb-
 capture-postrotate.sh’ ?
Is the shared config stored at ’/var/elbrus/shared/.config’ ? (y/n/exit)
 y
#global
SHAREDCONFIG=/var/elbrus/shared/.config

#paths
POSTROTATESCRIPT=/var/elbrus/capture/elb-capture-postrotate.sh
LOGFILE=/var/elbrus/shared/log

#settings
TIMEPERCAPTURE=900
MAXFILES=10
INTERFACE=eth0
Cleaning up...
elbrus@server:~/var/elbrus$
```

3.3.2 2 - Ohne Setup Script

Listing 44: Anhand von ’.env.example’ eigene ’.env’ Datei anlegen

```
#global
SHAREDCONFIG=/var/elbrus/shared/.config

#path
POSTROTATESCRIPT=/var/elbrus/capture/elb-capture-postrotate.sh
LOGFILEDIR=/var/elbrus/capture/capture"-$(date +"%Y-%U")".log

#settings
TIMEPERCAPTURE=900
MAXFILES=10
INTERFACE=eth0
```

3.4 Der Systemd Service

Listing 45: capture.service.example - Die Variable 'WorkingDirectory', Die Variable 'User' sowie die Variable 'ExecStopPost' anpassen.

```
...
#job is starting immediatly after the start action has been called
Type=simple
#the user to execute the script
User=aragog
#the working directory
WorkingDirectory=/var/elbrus/capture
#which script should be executed
ExecStart=/bin/bash elb-capture.sh
#when the script should restart
Restart=on-failure
#set the restart timeout
RestartSec=5
#which script should be executed when the service stops
ExecStopPost=/bin/bash elb-capture-log.sh

[Install]
...
```

Listing 46: Kopieren des Serviceprogrammes

```
elbrus@server:~/var/elbrus$ cp capture/capture.service.example\
/etc/systemd/system/capture.service
```

Listing 47: Neuladen des 'systemctl' Deamons

```
elbrus@server:~/var/elbrus$ systemctl daemon-reload
```

Listing 48: Aktivieren des Serviceprogrammes

```
elbrus@server:~/var/elbrus$ systemctl enable capture.service
```

Listing 49: Starten des Serviceprogrammes

```
elbrus@server:~/var/elbrus$ systemctl start capture.service
```

4 Packet Capture Importer

4.1 Installation

Listing 50: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/database.git
```

4.2 Umgebung Konfigurieren

4.2.1 1 - Mit Setup Script

Listing 51: Ausführen des setup Scripts

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ bash database/importer/pcap-importer/\
install.sh
Do you want to proceed? (y/n) y

Where is the shared config stored [/var/elbrus/shared/.config]: /var/
 elbrus/shared/.config
Where is the ’pcap-importer’ (dir) stored [/var/elbrus/pcap-importer]: /
 var/elbrus/pcap-importer

Would you like to store the ’pcap-importer’ at ’/var/elbrus/pcap-importer
 ’ ?
Is the shared config stored at ’/var/elbrus/shared/.config’ ? (y/n/exit)
 y
Submodule ’importer/pcap-importer/pcap-analyzer’ (https://github.com/
 rusticata/pcap-analyzer.git) registered for path ’importer/pcap-importer
 /pcap-analyzer’
Cloning into ’/var/elbrus/database/importer/pcap-importer/pcap-analyzer
 ’...
Submodule path ’importer/pcap-importer/pcap-analyzer’: checked out ’26
 abc0b0f4d9b2f0e6a72a62e694cd60ae6b6011’
Start Building ... (this may take a while)
Compiling proc-macro2 v1.0.38
Compiling unicode-xid v0.2.3
Compiling syn v1.0.93
...
Compiling libpcap-tools v0.1.0 (/var/elbrus/database/importer/pcap-
 importer/pcap-analyzer/libpcap-tools)
Compiling tokio-postgres v0.7.6
Compiling pcap-importer v0.1.0 (/var/elbrus/database/importer/pcap-
 importer)
Finished release [optimized] target(s) in 1m 38s
Cleaning up...
elbrus@server:~/var/elbrus$
```

4.2.2 2 - Ohne Setup Script

Listing 52: pcap-importer/.env - Anpassen an eigene Werte.

```
#where the traces should be stored
PCAPFOLDER=/var/elbrus/shared/traces/

#where the importer should be stored
IMPORTERPATH=/var/elbrus/pcap-importer

#database values
DB_HOST=10.0.76.220
DB_PORT=5432
DB_NAME=elbrus
DB_USER=elbrus
DB_PASSWORD=elbrus123!
```

5 Report Generator

5.1 Installation

Listing 53: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/report-generator.git
```

5.2 Umgebung Konfigurieren

5.2.1 1 - Mit Setup Script

Listing 54: Ausführen des ’install.sh’ Scripts.

```
elbrus@server:~/var/elbrus$ bash report-generator/install.sh
Do you want to proceed with setup of the ’report-generator’? (y/n) y

Where is the shared config stored [/var/elbrus/shared/.config]:

Is the shared config stored at ’/var/elbrus/shared/.config’ ? (y/n/exit) y
Install dependencies ...

...

elbrus@server:~/var/elbrus$
```

6 SNMP Manager

6.1 Installation

Listing 55: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/snmp-manager.git
```

6.2 Umgebung Konfigurieren

6.3 Der Systemd Service

Listing 56: snmp-manager.service.example - Die Variable ’WorkingDirectory’ sowie die Variable ’User’ anpassen.

```
...
#job is starting immediatly after the start action has been called
Type=simple
#the user to execute the script
User=elbrus
#the working directory
WorkingDirectory=/var/elbrus/snmp-manager/src
#which script should be executed
ExecStart=/bin/bash elb-snmp-manager.sh
...
```

Listing 57: Kopieren des Serviceprogrammes

```
elbrus@server:~/var/elbrus$ cp snmp-manager/src/snmp-manager.service\
.example /etc/systemd/system/snmp-manager.service
```

Listing 58: Kopieren des Zeitplanungsprogrammes.

```
elbrus@server:~/var/elbrus$ cp snmp-manager/src/snmp-manager.timer\
.example /etc/systemd/system/snmp-manager.timer
```

Listing 59: Neuladen des ’systemctl’ Deamons

```
elbrus@server:~/var/elbrus$ systemctl daemon-reload
```

Listing 60: Aktivieren des Serviceprogrammes

```
elbrus@server:~/var/elbrus$ systemctl enable snmp-manager.service
```

Listing 61: Aktivieren des Zeitplanungsprogrammes

```
elbrus@server:~/var/elbrus$ systemctl enable snmp-manager.timer
```

Listing 62: Starten des Zeitplanungsprogrammes

```
elbrus@server:~/var/elbrus$ systemctl start snmp-manager.timer
```

7 SSH Manager

7.1 Umgebung Konfigurieren

Kopieren von ’requirements.txt’, ’.env.example’, ’initialise.sh’, ’routine.sh’, ’setup.sh’, ’main.py’, ’ssh-manager.service.example’, ’ssh-manager-schedule.timer.example’ in den selben beliebigen Ordner.

Listing 63: Anhand von ’.env.example’ eigene ’.env’ Datei anlegen

```
#values regarding the jumpserver:
#IP, PORT and USER values must be set!
#depending on the usage you can set either:
#   -PASS and KEYFILE: keyfile is used with passphrase
#   -only KEYFILE: the keyfile is used
#   -only PASS: the password is used as is regular credentials
JUMPSERVER_IP="2.2.2.15"
JUMPSERVER_PORT=22
JUMPSERVER_USER=admin
JUMPSERVER_PASS=password
SSH_KEYFILE='my/sample/path'

#all database values must be set!
POSTGRES_HOST="192.168.0.1"
POSTGRES_PORT=245
POSTGRES_DB=mydb
POSTGRES_USER=admin
POSTGRES_PASS=password

#paths are configured by running 'setup.sh'
CONFIGPATH="/thats/where/i/store/my/configs"
MAINPATH="/the/path/to/main.py"
```

Listing 64: ssh-manager.service.example - Die Variable ’WorkingDirectory’ sowie die Variable ’User’ anpassen.

```
...
#job is starting immediatly after the start action has been called
Type=simple
#the user to execute the script
User=elbrus
#the working directory
WorkingDirectory=/home/elbrus/Desktop/ssh-manager/src/
#which script should be executed
ExecStart=/bin/bash routine.sh
...
```

7.1.1 1 - Mit Setup Script

Listing 65: Ausführen des setup Scripts

```
elbrus@server:~$ cd ssh-manager/src
elbrus@server:~/ssh-manager/src$ ./setup.sh
Setup for ssh-manager
Do you want to proceed? (y/n) y
we will proceed

Where do you want the config to be stored: (abolut path) /my/sample/path
Where is the ’main.py’ file stored: (abolut path) /path/to/main.py

Do you want to store the config files at "/my/sample/path"?
Is your ’main.py’ stored at "/path/to/main" (y/n/exit) y

The paths have been set!

Do you want to configure the systemd Service? (y/n/exit) y

Which User should execute the Service? elbrus

The systemd Service has been configured!

Do you want to run the initialise script? (y/n/exit) y

...

finished setup

elbrus@server:~/ssh-manager/src$
```

7.1.2 2 - Ohne Setup Script

Listing 66: .env - Die Variable ’CONFIGPATH’ sowie die Variable ’MAINPATH’ anpassen.

```
...
POSTGRES_PASS=password

#paths are configured by running ’setup.sh’
CONFIGPATH="/thats/where/i/store/my/configs"
MAINPATH="/the/path/to/main.py"
```

Listing 67: Ausführen des Scripts zur Initialisierung des VCS Verzeichnisses.

```
elbrus@server:~$ ssh-manager/src/initialise.sh
```

7.2 Voraussetzungen

Listing 68: Installieren von fehlenden python3 Packages.

```
elbrus@server:~$ pip3 install -r ssh-manager/requirements.txt
```

7.3 Der Systemd Service

Listing 69: Kopieren des Serviceprogrammes

```
elbrus@server:~$ cp src/ssh-manager.service.example\
/etc/systemd/system/ssh-manager.service
```

Listing 70: Kopieren des Zeitplanungsprogrammes.

```
elbrus@server:~$ cp src/ssh-manager-schedule.timer.example\
/etc/systemd/system/ssh-manager-schedule.timer
```

Listing 71: Neuladen des 'systemctl' Deamons

```
elbrus@server:~$ systemctl daemon-reload
```

Listing 72: Aktivieren des Serviceprogrammes

```
elbrus@server:~$ systemctl enable ssh-manager.service
```

Listing 73: Aktivieren des Zeitplanungsprogrammes

```
elbrus@server:~$ systemctl enable ssh-manager-schedule.timer
```

Listing 74: Starten des Zeitplanungsprogrammes

```
elbrus@server:~$ systemctl start ssh-manager-schedule.timer
```

8 API

8.1 Installation

Listing 75: Clonen der Software von GitHub.

```
elbrus@server:~$ cd /var/elbrus
elbrus@server:~/var/elbrus$ git clone https://github.com/\
Elbrus-Analytics/api.git
elbrus@server:~/var/elbrus$ cd api
elbrus@server:~/var/elbrus/api$
```

8.2 Voraussetzungen

Listing 76: Nachinstallieren der Abhängigkeiten.

```
elbrus@server:~/var/elbrus/api$ npm install
```

Listing 77: Installieren von ’pm2’.

```
elbrus@server:~/var/elbrus/api$ npm install -g pm2
```

8.3 Umgebung Konfigurieren

Listing 78: Anhand von '.env.example' eigene '.env' Datei anlegen

```
# Application Name
APP_NAME=Elbrus-API

# Port number
PORT=3000

# BASE URL
BASE=https://localhost:3000

# URL of DB
DB_USER=
DB_HOST=
DB_DATABASE=
DB_PASSWORD=
DB_PORT=

# JWT
JWT_SECRET=thisisasamplesecret
JWT_ACCESS_EXPIRATION_MINUTES=30
JWT_REFRESH_EXPIRATION_DAYS=30

# SMTP configuration options for the email service
SMTP_HOST=
SMTP_PORT=
SMTP_USERNAME=
SMTP_PASSWORD=
EMAIL_FROM=
EMAIL_NAME=
```

8.4 Inbetriebnahme

Listing 79: Starten der API.

```
elbrus@server:~/var/elbrus/api$ pm2 start ecosystem.config.json
```

Die API läuft in folge automatisch im Hintergrund.

9 Webinterface