Spark mit Windows 10

https://phoenixnap.com/kb/install-spark-on-windows-10

Anleitung zur Installation von Apache Spark auf Windows 10. Dies bildet die Basis für den Einfachen Einsatz und Nutzung durch Sprachen wie Kotlin oder C# (Auch das geht, wie man hier sieht: https://docs.microsoft.com/de-de/dotnet/spark/what-is-apache-spark-dotnet)

Für die Nutzung von Spark bedarf es mindestens vier Vorgaben:

1. Java muss in einer neueren Version installiert sein
2. Python muss in einer neueren Version installiert sein
3. Apache Spark selbst muss installiert sein
4. Zudem muss die Zusammenarbeit mit Hadoop berücksichtigt werden

Hinweis: In den Abbildungen tauchen zum Teil unterschiedliche Versionen der Software auf. Dies ist darauf zurückzuführen, dass ich an zwei Rechnern jeweils unterschiedliche Versionen installiert habe und die Screenshots während dieser beiden Installationen entstanden.

Java

Mit Hilfe des Befehls java -version kann die aktuell installierte Version überprüft werden. Gleichzeitig ist so sichergestellt, dass die Systemvariablen für Java korrekt eingestellt wurden:

```
PS C:\> java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) Client VM (build 25.261-b12, mixed mode)
PS C:\>
```

Ist Java auf dem System installiert erscheint eine Meldung ähnlich der oben zu sehenden. Hier im Beispiel ist die Version 1.8 installiert, welche aktuell ist.

Sollte dies nicht der Fall sein, so kann unter https://java.com/en/download/ eine aktuelle Version installiert werden. Hier findet man auch die entsprechenden Schritte zur Installation.

Python

Mit Hilfe des Befehls python --version kann die Version der installierten Python Version abgerufen werden. Auch hier signalisiert der erfolgreiche Aufruf, dass die Umgebungsvariablen gesetzt wurden.

```
PS C:\> python --version
Python 3.9.6
PS C:\>
```

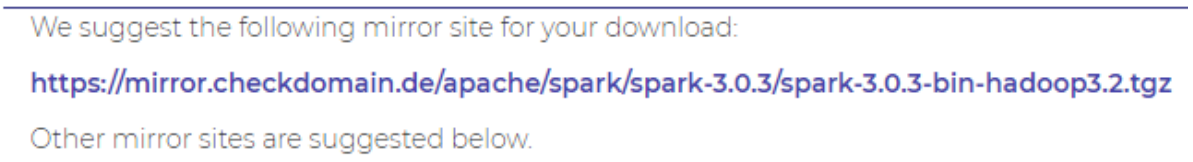
In diesem Fall ist die aktuelle Version 3.9.6 vorhanden. Für den Fall, dass Python nicht installiert ist, können die notwendigen Installationsdatei und Anleitung auf der Seite von https://www.python.org abgerufen werden. Der Download selbst befindet sich beispielsweise hier: https://www.python.org/downloads/release/python-396/ (hier für die Version 3.9.6).

Spark

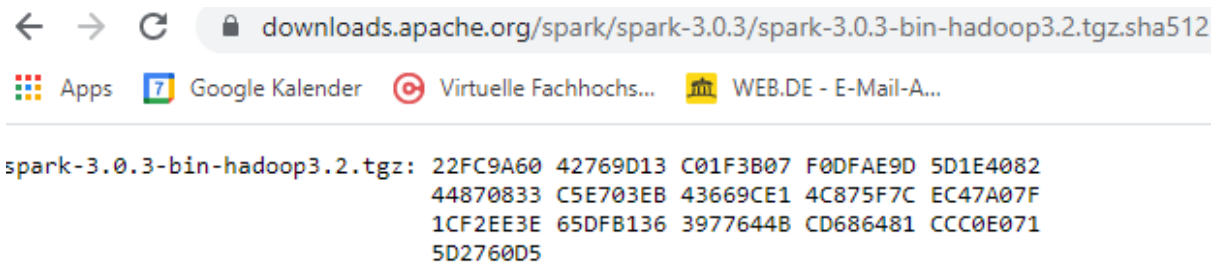
https://spark.apache.org/downloads.html

Die notwendigen Dateien finden sich an der oben genannten Adresse. Hier kann unter anderen die Release Version sowie der Typ eingestellt werden. An dieser Stelle wird die Auswahl wie nachfolgend zu sehen getroffen:

Über den an Punkt 3 gezeigten Link gelangt man zu einer Seite mit verschiedenen Mirrors, wobei der oberste in der Regel eine gute Wahl darstellt.



Nach dem Download kann die heruntergeladene Datei mit Hilfe der Checksumme verifiziert werden. Hierzu folgt man den unter Punkt 4 angegebenen Link zur Checksumme:

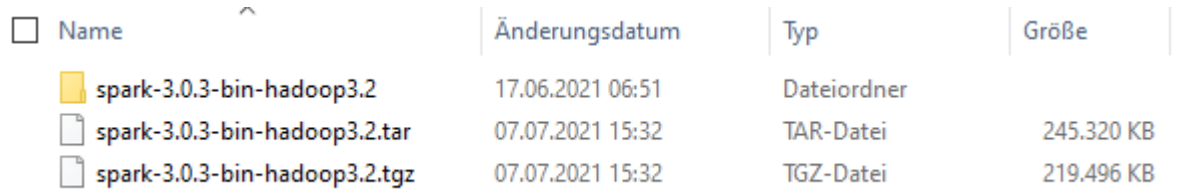


Mit dem Befehl ***certutil*** kann nun die Checksumme aus der heruntergeladenen Datei generiert und mit der angezeigten verglichen werden. Hierbei sollte darauf geachtet werden, dass die Versionen zueinander passen, da es leicht zu Verwechslungen kommen kann.

```
PS C:\> certutil -hashfile c:\users\username\Downloads\spark-3.0.3-bin-h
adoop3.2.tgz SHA512
CertUtil: -hashfile-Befehl ist fehlgeschlagen: 0x80070002 (WIN32: 2 ERRO
R_FILE_NOT_FOUND)
CertUtil: Das System kann die angegebene Datei nicht finden.
PS C:\> certutil -hashfile C:\Users\ckitte\Downloads\spark-3.0.3-bin-had
oop3.2.tgz SHA512
SHA512-Hash von C:\Users\ckitte\Downloads\spark-3.0.3-bin-hadoop3.2.tgz:

22fc9a6042769d13c01f3b07f0dfae9d5d1e408244870833c5e703eb43669ce14c875f7c
ec47a07f1cf2ee3e65dfb1363977644bcd686481ccc0e0715d2760d5
CertUtil: -hashfile-Befehl wurde erfolgreich ausgeführt.
PS C:\>
```

Die Installation von Spark ist eigentlich ein Entpacken der heruntergeladenen Datei in ein frei zu bestimmendes Verzeichnis. Für das Entpacken kann das normale und in Windows vorhandene ZIP Tool verwendet werden.

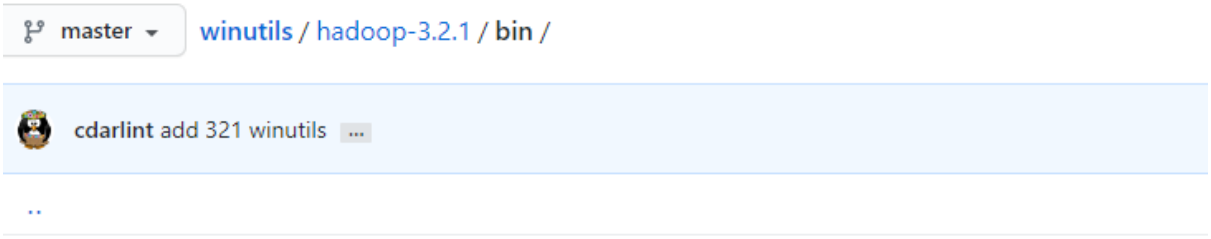
Nach dem Entpacken der tgz Datei, existiert eine neue Datei mit der Endung tar. Hierbei handelt es sich um ein gezipptes Verzeichnis. Nachdem diese Datei ebenfalls entpackt wurde, liegt das eigentliche Verzeichnis vor:



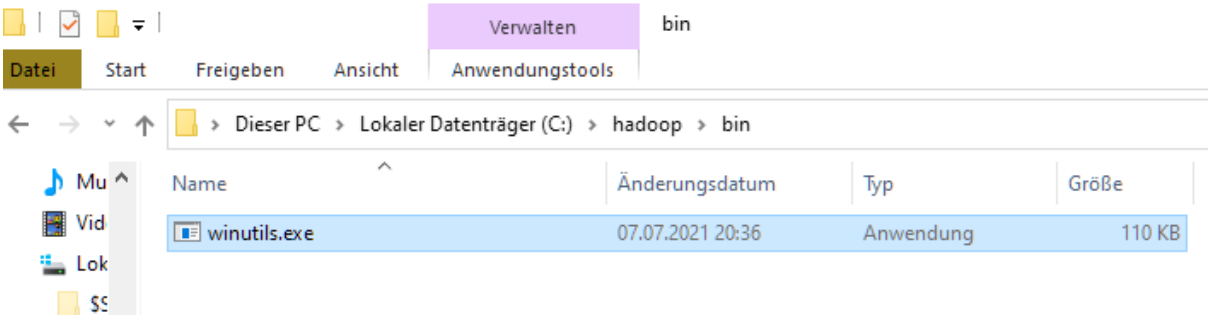
In dem angelegten Ordner befinden sich alle zu Spark gehörenden Dateien, welche nun verfügbar sind.

Spark baut auf Hadoop auf. die hier verwendete Version 3.2 wurde beim Download (siehe oben) ausgewählt. Zum Betrieb unter Windows muss jedoch keine komplette Hadoop Umgebung realisiert werden. Allerdings wird unter Windows ein kleines Tool benötigt, da Hadoop Probleme mit dem Dateiverzeichnis von Windows hat.

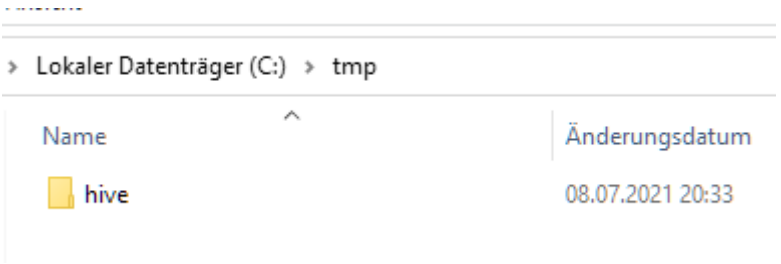
Das Tool kann unter https://github.com/cdarlint/winutils passend für die gewählte Version herunter geladen werden.



Hierzu wird in das passende Verzeichnis auf GitHub gegangen und die Datei winutils.exe heruntergeladen. Diese wird anschließend in einem Verzeichnis c:\hadoop\bin hinein kopiert.



Weiter benötigt Hadoop nach Aussage einiger Quellen für das korrekte Funktionieren ein temporäres Verzeichnis unter c:\tmp\hive zur Speicherung von Zwischenwerte. Diese Aussage konnte jedoch nicht sicher geprüft werden.



Das angelegte Verzeichnis muss für winutil verfügbar sein. Dies erreicht man durch das Ausführen des folgenden Kommandos:

```
PS C:\Users\Christian Kitte> winutils chmod 777 c:\tmp\hive
PS C:\Users\Christian Kitte>
```

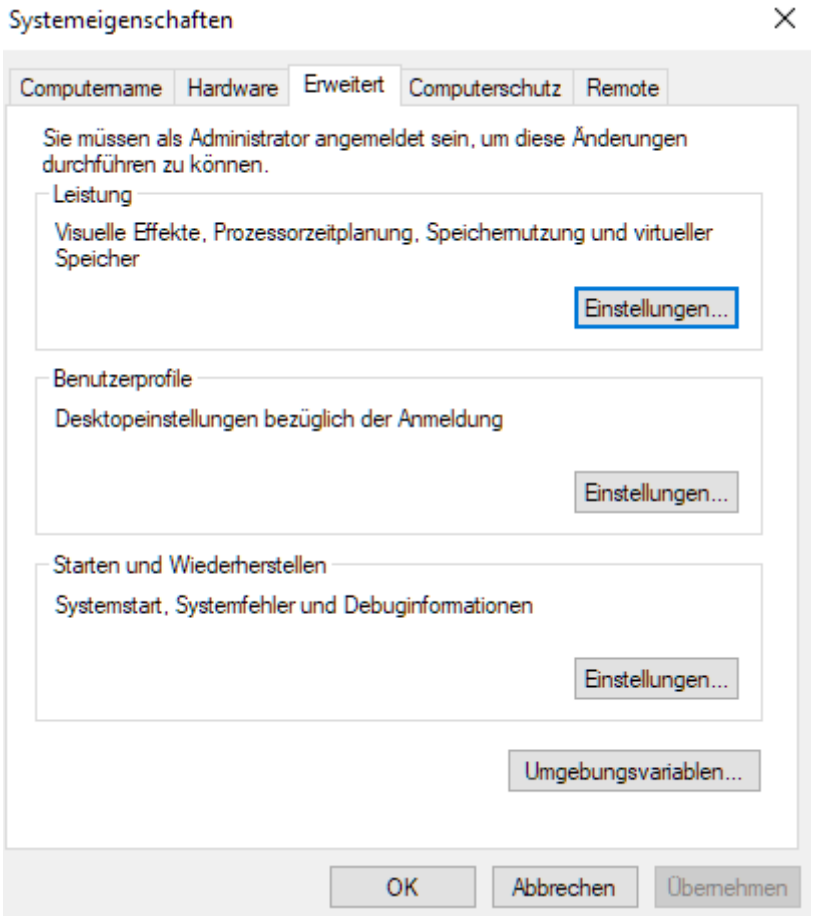
Abschließend müssen im nächsten Schritt noch einige Pfade angepasst werden, damit alle Komponenten zusammenarbeiten können. Die folgenden Umgebungsvariablen müssen hierbei festgelegt, sowie die Pfadvariablen entsprechend erweitert werden:

SPARK_HOME
c:\spark\spark-3.1.3-bin-hadoop3.2
Path ergänzen mit %SPARK_HOME%\bin

HADOOP_HOME
C:\hadoop
Path ergänzen mit %HADOOP_HOME%\bin

JAVA_HOME
c:\Program Files (x86)\Java\jre1.8.0_291 (jre reicht)
keine Pfad

Für die Umsetzung wechselt man über ***System*** in ***Systemeigenschaften*** und hier in die Bearbeitung der ***Umgebungsvariablen*** (Button unten rechts auf dem Reiter “Erweitert”):



Im Anschluss sollten alle Umgebungsvariablen verfügbar sein (hier in Powershell):

```
PS C:\> $env:SPARK_HOME
c:\spark\spark-3.1.2-bin-hadoop3.2
PS C:\> $env:HADOOP_HOME
c:\hadoop
PS C:\> $env:JAVA_HOME
C:\Program Files (x86)\Java\jre1.8.0_291
PS C:\>
```

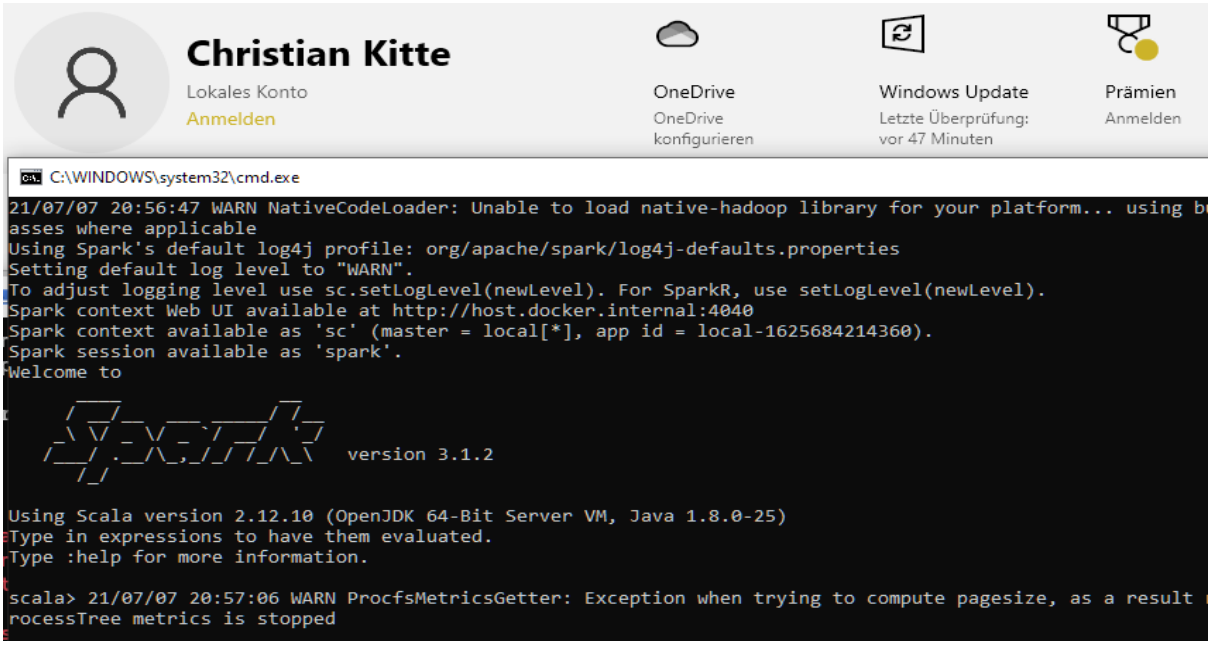
und hier in der normalen Eingabeaufforderung:

```
C:\Users\Christian Kitte>echo %SPARK_HOME%
c:\spark\spark-3.1.2-bin-hadoop3.2

C:\Users\Christian Kitte>echo %HADOOP_HOME%
c:\hadoop

C:\Users\Christian Kitte>echo %JAVA_HOME%
C:\Program Files (x86)\Java\jre1.8.0_291
```

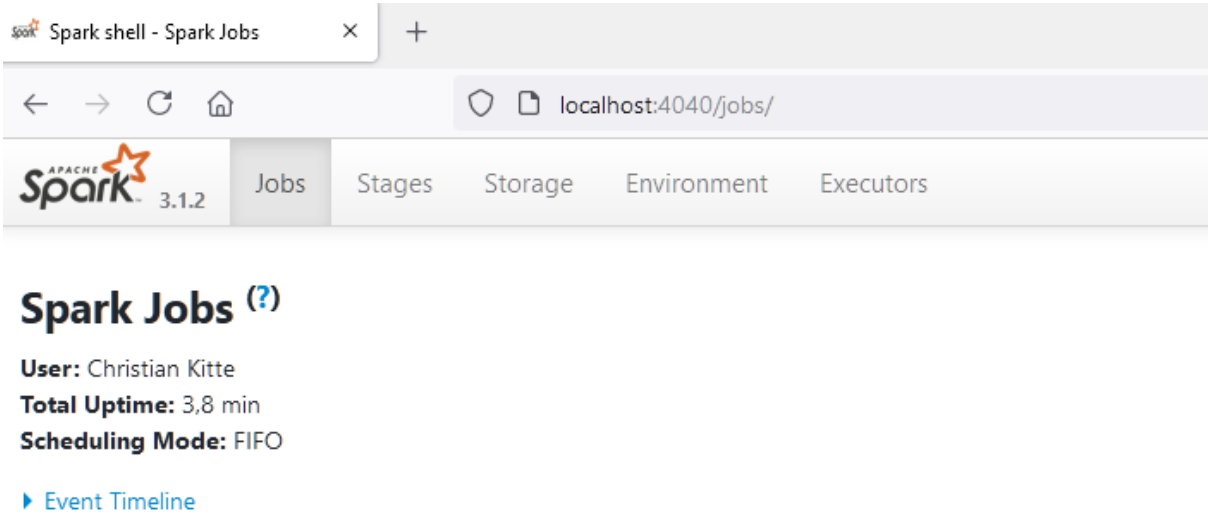
Somit ist nunmehr die Installation abgeschlossen. mit dem Befehl **spark-shell** lässt sich die Konsole von Spark aufrufen;

```
PS C:\> C:\spark\spark-3.1.2-bin-hadoop3.2\bin\spark-shell
PS C:\>
```

Nach der Bestätigung öffnet sich eine neue Konsole und Spark startet mit einer Reihe von Ausgaben.



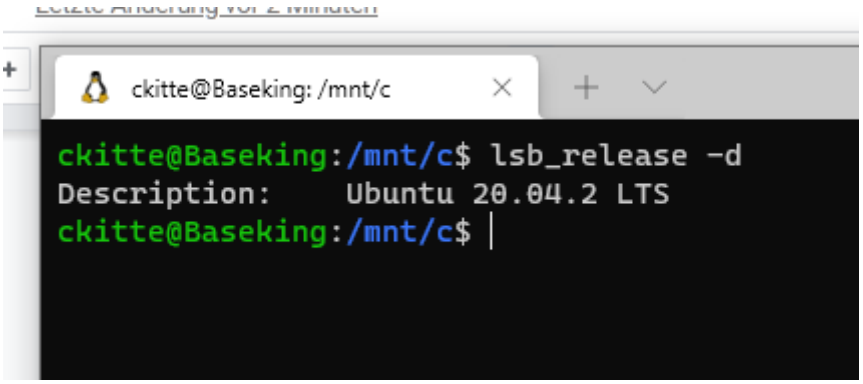
Gleichzeitig ist die shell Web UI von Spark über den Browser am standardmäßigen Port 4040 erreichbar:



Spark mit dem WSL

WSL steht für Windows Subsystem für Linux und beinhaltet eine komplette Umgebung für Linux. Dies ermöglicht z.B. den einfachen und nativen Umgang mit Docker Container für Linux ohne Umwege.

Das WSL kann problemlos erweitert werden. So habe ich beispielsweise ein komplettes Ubuntu System eingerichtet:



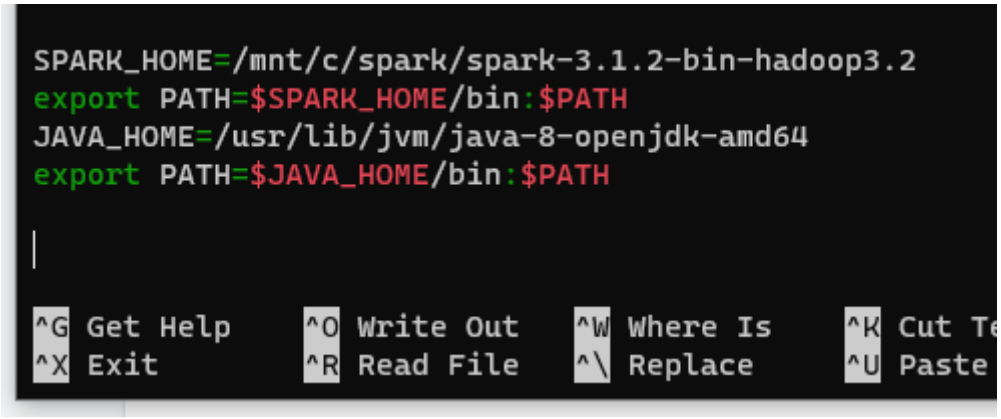
Im folgenden habe ich auf diesem System zunächst Java, dann Python sowie abschließend PySpark separat installiert:

sudo apt-get -y install openjdk-8-jdk-headless
sudo apt install python3
sudo apt install pyspark

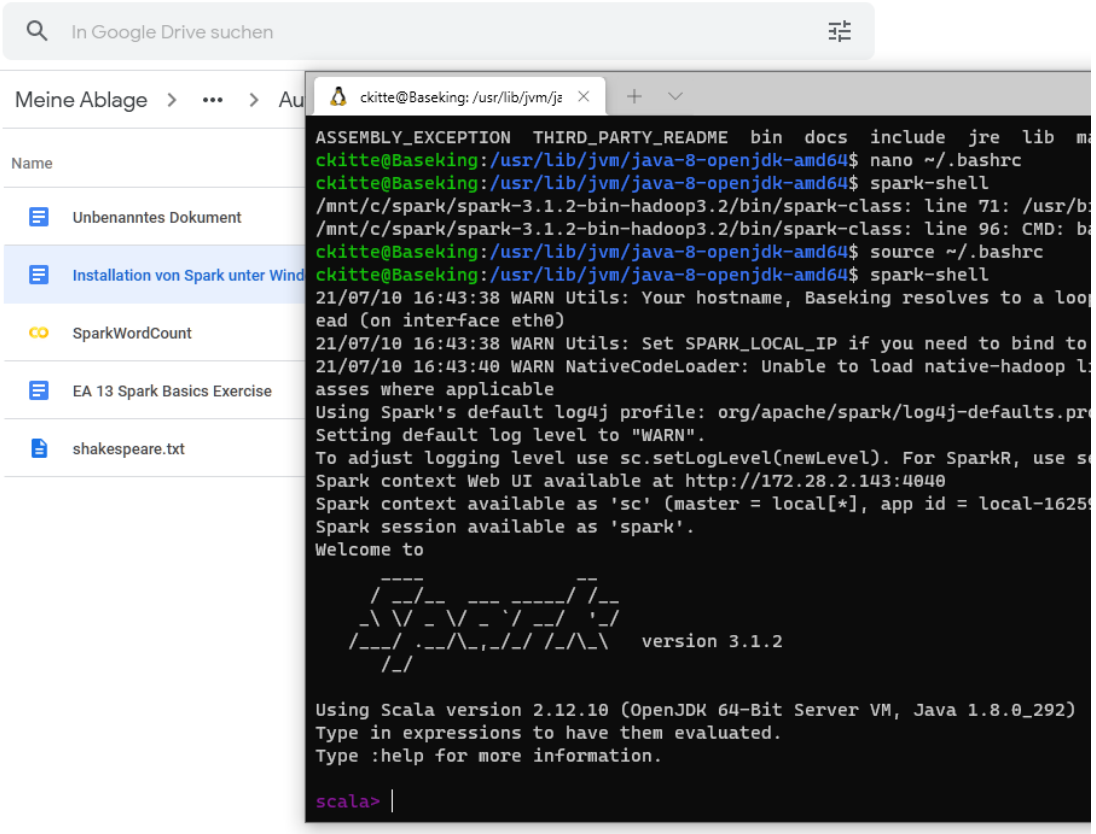
Obwohl die Pfade der Anleitung entsprechend gesetzt waren, kam es wie zuvor bei Windows selbst und entgegen der Vorhersage der Anleitung, zu Fehlern beim Aufruf. Entweder wurden Java Klassen nicht gefunden, oder Skripte wie spark-submit, welcher definitiv vorhanden waren.

Letztlich stellte es sich heraus, dass zumindest im WSL das Problem bei den Pfaden lag. Letztlich setzt ich die Umgebungsvariablen wie folgendermaßen:

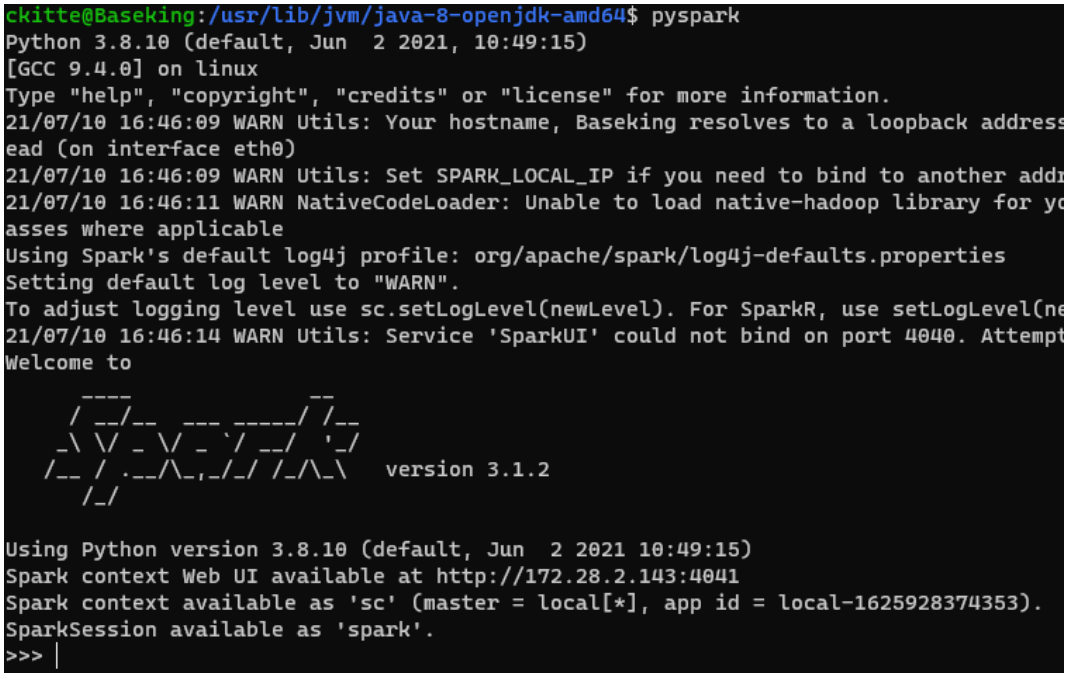
(*Exkurs: Pfade können unter Linux mit Hilfe der Datei **bashrc** definiert werden. Ich verwende hier gerne **nano** als einfachen Editor. Mit **nano ~/.bashrc** wird die Datei geöffnet. Nach dem Speichern und schließen müssen die Änderungen bekannt gemacht werden mit **source ~/.bashrc**.*)



SPARK_HOME zeigt hierbei auf das auch für Windows verwendete Verzeichnis ohne den Anhang **bin**. Hierbei verwende ich den **vollen Pfad**. Ebenso verwende ich für **JAVA_HOME** den **vollen Pfad**. Als Ergebnis konnte ich Spark unter Windows innerhalb des Windows Subsystems für Linux sowohl für Scala:



als auch für Python:



starten.