Einführung in Cloud Computing und Big Data

Diese Lektion behandelt die folgenden Themen:

- Spark
- Hadoop
- Kubernetes
- Docker
- Flink
- MongoDB

Wenn es um Cloud Computing geht, scheint es unvermeidlich, viele Tools zu erwähnen: Hadoop, Hive, Hbase, ZooKeeper, Docker, Kubernetes, Spark, Kafka, MongoDB, Flink, Druid, Presto, Kylin, Elastic Search. Haben Sie schon von all diesen gehört? Einige dieser Tools habe ich aus den Stellenbeschreibungen von `Big Data Engineers` und `Distributed Backend Engineers` gefunden. Dies sind alles gut bezahlte Positionen. Versuchen wir, sie alle zu installieren und ein wenig damit zu spielen. ## Erste Schritte mit Spark

Die offizielle Website besagt, dass `Spark` ein Analyse-Engine zur Verarbeitung von Massendaten ist. `Spark` ist im Wesentlichen eine Sammlung von Bibliotheken. Es scheint nicht wie `Redis` in Server- und Client-Komponenten unterteilt zu sein. `Spark` wird ausschließlich auf der Client-Seite verwendet. Von der offiziellen Website habe ich die neueste Version heruntergeladen, `spark-3.1.1-bin-hadoop3.2.tar`.

```
$ tree . -L 1
.
  LICENSE
  NOTICE
  R
  README.md
  RELEASE
  bin
  conf
  data
  examples
  jars
  kubernetes
```

```
  licenses
  python
  sbin
  yarn
```

11 Verzeichnisse, 4 Dateien

```
Es scheint sich um einige Analysebibliotheken zu handeln, die in verschiedenen Sprachen geschrieben sin
```

```
Gleichzeitig sagt die offizielle Website, dass man die Abhängigkeiten direkt in Python installieren kan
```

```shell
$ pip install pyspark
Collecting pyspark
  Downloading pyspark-3.1.1.tar.gz (212,3 MB)
     |                 | 212,3 MB 14 kB/s
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     |                 | 198 kB 145 kB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.1-py2.py3-none-any.whl size=212767604 sha256=0b8079e8
  Stored in directory: /Users/lzw/Library/Caches/pip/wheels/23/bf/e9/9f3500437422e2ab82246f25a51ee480a4
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.1
```

Installiert.

Ich habe mir die offizielle Website angesehen und einige Beispiele gefunden.

```
./bin/run-example SparkPi 10
```

Oh, ich sehe, du kannst das Programm aus dem gerade heruntergeladenen Installationspaket ausführen, aber es ist ein Fehler aufgetreten.

```
$ ./bin/run-example SparkPi 10
21/03/11 00:06:15 WARN NativeCodeLoader: Native-Hadoop-Bibliothek konnte für Ihre Plattform nicht gelad
```

```
21/03/11 00:06:16 INFO ResourceUtils: Keine benutzerdefinierten Ressourcen für spark.driver konfiguriert
21/03/11 00:06:16 WARN Utils: Der Dienst 'sparkDriver' konnte keinen zufälligen freien Port binden. Über
```

> Spark ist eine schnelle und vielseitige Verarbeitungs-Engine, die mit Hadoop-Daten kompatibel ist. Es kann in Hadoop-Clustern über YARN oder im eigenständigen Modus von Spark ausgeführt werden und kann Daten in HDFS, HBase, Cassandra, Hive und jedem Hadoop InputFormat verarbeiten. Es wurde entwickelt, um sowohl Batch-Verarbeitung (ähnlich wie MapReduce) als auch neue Workloads wie Streaming, interaktive Abfragen und maschinelles Lernen zu unterstützen.

Es ist mehrmals `hadoop` aufgetaucht. Nachdem ich `spark depends hadoop` gegoogelt habe, bin ich auf folgenden Absatz gestoßen. Es scheint, dass dies von Daten im `Hadoop`-Format abhängt. Lassen Sie uns zunächst `Hadoop` untersuchen.

Hadoop

Nachdem ich die offizielle Website kurz überflogen habe, werde ich es jetzt installieren.

```
brew install hadoop
```

Während des Installationsprozesses lass uns etwas darüber lernen.

> Die Apache Hadoop-Softwarebibliothek ist ein Framework, das die verteilte Verarbeitung großer Datenmengen über Computercluster hinweg mithilfe einfacher Programmiermodelle ermöglicht. Es ist darauf ausgelegt, sich von einzelnen Servern auf Tausende von Maschinen zu skalieren, wobei jede lokale Berechnungen und Speicherung bietet. Anstatt sich auf Hardware zu verlassen, um Hochverfügbarkeit zu gewährleisten, ist die Bibliothek selbst so konzipiert, dass sie Fehler auf der Anwendungsebene erkennt und behandelt. Dadurch bietet sie einen hochverfügbaren Dienst auf Basis eines Computerclusters, bei dem jede einzelne Maschine anfällig für Ausfälle sein kann.

Hadoop ist ein Framework, das entwickelt wurde, um verteilte Datensätze zu verarbeiten. Diese Datensätze können auf vielen Computern verteilt sein. Es verwendet ein sehr einfaches Programmiermodell. Es ist darauf ausgelegt, sich von einem einzelnen Server auf Tausende von Maschinen zu skalieren. Anstatt sich auf die hohe Verfügbarkeit von Hardware zu verlassen, ist diese Bibliothek so konzipiert, dass sie Fehler auf der Anwendungsebene erkennen

und behandeln kann. Dadurch kann ein hochverfügbarer Dienst in einem Cluster bereitgestellt werden, obwohl jede einzelne Maschine im Cluster potenziell ausfallen könnte.

```
$ brew install hadoop
Fehler:
  homebrew-core ist ein flacher Klon.
  homebrew-cask ist ein flacher Klon.
Um `brew update` auszuführen, führen Sie zuerst folgende Befehle aus:
  git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-core fetch --unshallow
  git -C /usr/local/Homebrew/Library/Taps/homebrew/homebrew-cask fetch --unshallow
Diese Befehle können einige Minuten dauern, da die Repositorys sehr groß sind.
Diese Einschränkung wurde auf Anfrage von GitHub vorgenommen, da das Aktualisieren von flachen Klonen au
==> Lade https://homebrew.bintray.com/bottles/openjdk-15.0.1.big_sur.bottle.tar.gz herunter
Bereits heruntergeladen: /Users/lzw/Library/Caches/Homebrew/downloads/d1e3ece4af1d225bc2607eaa4ce85a873
==> Lade https://www.apache.org/dyn/closer.lua?path=hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz herun
Bereits heruntergeladen: /Users/lzw/Library/Caches/Homebrew/downloads/764c6a0ea7352bb8bb505989feee1b36d
==> Installiere Abhängigkeiten für hadoop: openjdk
==> Installiere hadoop-Abhängigkeit: openjdk
==> Gieße openjdk-15.0.1.big_sur.bottle.tar.gz
==> Hinweise
Damit die System-Java-Wrapper dieses JDK finden, verlinken Sie es mit
  sudo ln -sfn /usr/local/opt/openjdk/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk.jdk
```

openjdk ist keg-only, was bedeutet, dass es nicht nach /usr/local symbolisch verlinkt wurde, weil es den macOS `java`-Wrapper überschattet.

Wenn Sie openjdk zuerst in Ihrem PATH haben müssen, führen Sie den folgenden Befehl aus:

```
bash   echo 'export PATH="/usr/local/opt/openjdk/bin:$PATH"' >> /Users/lzw/.bash_profile
```

Damit Compiler OpenJDK finden können, müssen Sie möglicherweise Folgendes setzen: `bash`

```
export CPPFLAGS="-I/usr/local/opt/openjdk/include"
```

==> Zusammenfassung /usr/local/Cellar/openjdk/15.0.1: 614 Dateien, 324,9 MB ==> Installiere hadoop /usr/local/Cellar/hadoop/3.3.0: 21.819 Dateien, 954,7 MB, gebaut in 2 Minuten 15 Sekunden ==> Aktualisiere 1 Abhängigkeit: maven 3.3.3 -> 3.6.3_1 ==> Aktualisiere maven 3.3.3 -> 3.6.3_1 ==> Lade https://www.apache.org/dyn/closer.lua?path=maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz herunter ==> Lade von https://mirror.olnevhost.net/pub/apac 3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz herunter ## 100,0% Fehler: Der `brew link` Schritt wurde nicht erfolgreich abgeschlossen Die Formel wurde

gebaut, aber nicht in /usr/local verlinkt Konnte bin/mvn nicht verlinken Ziel /usr/local/bin/mvn ist ein Symlink, der zu maven gehört. Sie können ihn entlinken: brew unlink maven

Um den Link zu erzwingen und alle konfligierenden Dateien zu überschreiben: brew link –overwrite maven

Um alle Dateien aufzulisten, die gelöscht würden: brew link –overwrite –dry-run maven

Mögliche konfliktträchtige Dateien sind: /usr/local/bin/mvn -> /usr/local/Cellar/maven/3.3.3/bin/mvn /usr/local/bin/mvnDebug -> /usr/local/Cellar/maven/3.3.3/bin/mvnDebug /usr/local/bin/mvnyjp -> /usr/local/Cellar/maven/3.3.3/bin/mvnyjp ==> Zusammenfassung /usr/local/Cellar/maven/3.6.3_1: 87 Dateien, 10,7MB, in 7 Sekunden erstellt Entferne: /usr/local/Cellar/maven/3.3.3... (92 Dateien, 9MB) ==> Überprüfe Abhängigkeiten der aktualisierten Formeln... ==> Keine defekten Abhängigkeiten gefunden! ==> Hinweise ==> openjdk Damit die System-Java-Wrapper dieses JDK finden, verlinken Sie es mit sudo ln -sfn /usr/local/opt/openjdk/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk.jdk

openjdk ist keg-only, was bedeutet, dass es nicht in /usr/local symbolisch verlinkt wurde, weil es den macOS `java`-Wrapper überschattet.

Wenn Sie sicherstellen möchten, dass openjdk in Ihrem PATH an erster Stelle steht, führen Sie den folgenden Befehl aus: `bash  echo 'export PATH="/usr/local/opt/openjdk/bin:$PATH"' >> /Users/lzw/.bash_profile`

Damit Compiler OpenJDK finden können, müssen Sie möglicherweise folgendes setzen: export CPPFLAGS=“-I/usr/local/opt/openjdk/include”

```
Beachten Sie, dass in den Ausgabelogs von `brew` `maven` nicht korrekt verlinkt wurde. Führen Sie als Na
```

```shell
  brew link --overwrite maven
```

`Hadoop` wurde erfolgreich installiert.

Module

Das Projekt umfasst folgende Module:

- **Hadoop Common**: Die allgemeinen Hilfsprogramme, die die anderen Hadoop-Module unterstützen.

- **Hadoop Distributed File System (HDFS™)**: Ein verteiltes Dateisystem, das einen hohen Durchsatz für den Zugriff auf Anwendungsdaten bietet.
- **Hadoop YARN**: Ein Framework für die Jobplanung und die Verwaltung von Clusterressourcen.
- **Hadoop MapReduce**: Ein YARN-basiertes System zur parallelen Verarbeitung großer Datenmengen.
- **Hadoop Ozone**: Ein Objektspeicher für Hadoop.

Es gibt diese Module. Wenn Sie `hadoop` eingeben, erscheint:

```
$ hadoop
Verwendung: hadoop [OPTIONEN] UNTERBEFEHL [UNTERBEFEHL-OPTIONEN]
 oder    hadoop [OPTIONEN] KLASSENNAME [KLASSENNAME-OPTIONEN]
  wobei KLASSENNAME eine vom Benutzer bereitgestellte Java-Klasse ist

OPTIONS ist keine oder eine der folgenden Optionen:

--config dir                     Hadoop-Konfigurationsverzeichnis
--debug                          Debug-Modus für Shell-Skripte aktivieren
--help                           Nutzungsinformationen
buildpaths                       Versuch, Klassendateien aus dem Build-Verzeichnis hinzuzufügen
hostnames list[,of,host,names]   Hosts, die im Slave-Modus verwendet werden sollen
hosts filename                   Liste der Hosts, die im Slave-Modus verwendet werden sollen
loglevel level                   Log4j-Level für diesen Befehl festlegen
workers                          Worker-Modus aktivieren

  SUBCOMMAND ist eines der folgenden:
    Admin-Befehle:

daemonlog     Protokollstufe für jeden Daemon abrufen/festlegen

    Client-Befehle:
```

archive Erstellt ein Hadoop-Archiv checknative Überprüft die Verfügbarkeit nativer Hadoop- und Kompressionsbibliotheken classpath Gibt den Klassenpfad aus, der benötigt wird, um das Hadoop-JAR und die erforderlichen Bibliotheken zu erhalten conftest Validiert Konfigurations-XML-Dateien credential Interagiert mit Anmeldeinformationsanbietern distch Verteilter Meta-

datenänderer distcp Kopiert Dateien oder Verzeichnisse rekursiv dtutil Operationen im Zusammenhang mit Delegationstokens envvars Zeigt die berechneten Hadoop-Umgebungsvariablen an fs Führt einen generischen Dateisystem-Client aus gridmix Sendet eine Mischung aus synthetischen Jobs, die ein Profil aus der Produktionslast modellieren jar Führt eine JAR-Datei aus. HINWEIS: Bitte verwenden Sie “yarn jar”, um YARN-Anwendungen zu starten, nicht diesen Befehl. jnipath Gibt den java.library.path aus kdiag Diagnostiziert Kerberos-Probleme kerbname Zeigt die auth_to_local Principal-Konvertierung an key Verwaltet Schlüssel über den KeyProvider rumenfolder Skaliert eine Rumen-Eingabespur rumentrace Konvertiert Protokolle in eine Rumen-Spur s3guard Verwaltet Metadaten auf S3 trace Zeigt und modifiziert Hadoop-Tracing-Einstellungen version Gibt die Version aus

```
    Daemon-Befehle:

kms           KMS ausführen, den Key Management Server
registrydns   den Registry-DNS-Server ausführen

SUBCOMMAND kann Hilfe anzeigen, wenn es ohne Parameter oder mit -h aufgerufen wird.
```

Die offizielle Website bietet einige Beispiele.

```
  $ mkdir input
  $ cp etc/hadoop/*.xml input
  $ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z
  $ cat output/*
```

Beachte, dass es `share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar` gibt. Dies bedeutet, dass möglicherweise einige Beispiel-Dateien fehlen, die wir nicht erhalten haben. Es wird vermutet, dass bei der Installation mit `Homebrew` diese Dateien nicht enthalten sind. Wir haben das Installationspaket von der offiziellen Website heruntergeladen.

```
$ tree . -L 1
.
   LICENSE-binary
   LICENSE.txt
   NOTICE-binary
   NOTICE.txt
   README.txt
```

```
  bin
  etc
  include
  lib
  libexec
  licenses-binary
  sbin
  share
```

Es scheint, dass ein `share`-Verzeichnis vorhanden ist. Aber hat `Homebrew` wirklich keine dieser zusätzlichen Dateien? Finde das Verzeichnis, in dem `Homebrew` installiert ist.

```
$ type hadoop
hadoop ist /usr/local/bin/hadoop
$ ls -alrt /usr/local/bin/hadoop
lrwxr-xr-x  1 lzw  admin  33 Mar 11 00:48 /usr/local/bin/hadoop -> ../Cellar/hadoop/3.3.0/bin/hadoop
$ cd /usr/local/Cellar/hadoop/3.3.0
```

Dies ist das Verzeichnisbaum, das unter `/usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop` gedruckt wurde.

```
$ tree . -L 2
.
  client
     hadoop-client-api-3.3.0.jar
     hadoop-client-minicluster-3.3.0.jar
     hadoop-client-runtime-3.3.0.jar
  common
     hadoop-common-3.3.0-tests.jar
     hadoop-common-3.3.0.jar
     hadoop-kms-3.3.0.jar
     hadoop-nfs-3.3.0.jar
     hadoop-registry-3.3.0.jar
     jdiff
     lib
     sources
     webapps
```

```
hdfs
    hadoop-hdfs-3.3.0-tests.jar
    hadoop-hdfs-3.3.0.jar
    hadoop-hdfs-client-3.3.0-tests.jar
    hadoop-hdfs-client-3.3.0.jar
    hadoop-hdfs-httpfs-3.3.0.jar
    hadoop-hdfs-native-client-3.3.0-tests.jar
    hadoop-hdfs-native-client-3.3.0.jar
    hadoop-hdfs-nfs-3.3.0.jar
    hadoop-hdfs-rbf-3.3.0-tests.jar
    hadoop-hdfs-rbf-3.3.0.jar
    jdiff
    lib
    sources
    webapps
mapreduce
    hadoop-mapreduce-client-app-3.3.0.jar
    hadoop-mapreduce-client-common-3.3.0.jar
    hadoop-mapreduce-client-core-3.3.0.jar
    hadoop-mapreduce-client-hs-3.3.0.jar
    hadoop-mapreduce-client-hs-plugins-3.3.0.jar
    hadoop-mapreduce-client-jobclient-3.3.0-tests.jar
    hadoop-mapreduce-client-jobclient-3.3.0.jar
    hadoop-mapreduce-client-nativetask-3.3.0.jar
    hadoop-mapreduce-client-shuffle-3.3.0.jar
    hadoop-mapreduce-client-uploader-3.3.0.jar
    hadoop-mapreduce-examples-3.3.0.jar
    jdiff
    lib-examples
    sources
tools
    dynamometer
    lib
    resourceestimator
    sls
    sources
```

```
yarn
    csi
    hadoop-yarn-api-3.3.0.jar
    hadoop-yarn-applications-catalog-webapp-3.3.0.war
    hadoop-yarn-applications-distributedshell-3.3.0.jar
    hadoop-yarn-applications-mawo-core-3.3.0.jar
    hadoop-yarn-applications-unmanaged-am-launcher-3.3.0.jar
    hadoop-yarn-client-3.3.0.jar
    hadoop-yarn-common-3.3.0.jar
    hadoop-yarn-registry-3.3.0.jar
    hadoop-yarn-server-applicationhistoryservice-3.3.0.jar
    hadoop-yarn-server-common-3.3.0.jar
    hadoop-yarn-server-nodemanager-3.3.0.jar
    hadoop-yarn-server-resourcemanager-3.3.0.jar
    hadoop-yarn-server-router-3.3.0.jar
    hadoop-yarn-server-sharedcachemanager-3.3.0.jar
    hadoop-yarn-server-tests-3.3.0.jar
    hadoop-yarn-server-timeline-pluginstorage-3.3.0.jar
    hadoop-yarn-server-web-proxy-3.3.0.jar
    hadoop-yarn-services-api-3.3.0.jar
    hadoop-yarn-services-core-3.3.0.jar
    lib
    sources
    test
    timelineservice
    webapps
    yarn-service-examples
```

Man kann viele `jar`-Pakete sehen.

```
$ mkdir input
$ ls
bin         hadoop-config.sh    hdfs-config.sh      libexec         sbin            yarn-config.sh
etc         hadoop-functions.sh input           mapred-config.sh    share
$ cp etc/hadoop/*.xml input
$ cd input/
$ ls
```

```
capacity-scheduler.xml  hadoop-policy.xml   hdfs-site.xml       kms-acls.xml        mapred-site.xml
core-site.xml       hdfs-rbf-site.xml   httpfs-site.xml     kms-site.xml        yarn-site.xml
$ cd ..
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z.]
JAR existiert nicht oder ist keine normale Datei: /usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop/ma
$
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar grep input output 'dfs[a-z.]
2021-03-11 01:54:30,791 WARN util.NativeCodeLoader: Native-Hadoop-Bibliothek konnte für Ihre Plattform n
2021-03-11 01:54:31,115 INFO impl.MetricsConfig: Eigenschaften aus hadoop-metrics2.properties geladen
2021-03-11 01:54:31,232 INFO impl.MetricsSystemImpl: Metrik-Snapshot-Intervall auf 10 Sekunden festgeleg
...
```

Folgen Sie dem Beispiel auf der offiziellen Website. Beachten Sie, dass in `bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input` die `jar`-Datei eine Versionsnummer enthält. Daher müssen wir diese durch unsere Version `3.3.0` ersetzen.

Ende des Logs:

```
2021-03-11 01:54:35,374 INFO mapreduce.Job:  map 100% reduce 100%
2021-03-11 01:54:35,374 INFO mapreduce.Job: Job job_local2087514596_0002 erfolgreich abgeschlossen
2021-03-11 01:54:35,377 INFO mapreduce.Job: Zähler: 30
    Dateisystem-Zähler
        FILE: Anzahl der gelesenen Bytes=1204316
        FILE: Anzahl der geschriebenen Bytes=3565480
        FILE: Anzahl der Lesevorgänge=0
        FILE: Anzahl der großen Lesevorgänge=0
        FILE: Anzahl der Schreibvorgänge=0
    Map-Reduce-Framework
        Map-Eingabedatensätze=1
        Map-Ausgabedatensätze=1
        Map-Ausgabebytes=17
        Map-Ausgabematerialisierte Bytes=25
        Eingabe-Split-Bytes=141
        Combine-Eingabedatensätze=0
        Combine-Ausgabedatensätze=0
        Reduce-Eingabegruppen=1
        Reduce-Shuffle-Bytes=25
```

```
        Reduce-Eingabedatensätze=1
        Reduce-Ausgabedatensätze=1
        Verlorene Datensätze=2
        Shuffled Maps=1
        Fehlgeschlagene Shuffles=0
        Zusammengeführte Map-Ausgaben=1
        GC-Zeit verstrichen (ms)=57
        Gesamter belegter Heap-Speicher (Bytes)=772800512
    Shuffle-Fehler
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    Datei-Eingabeformat-Zähler
        Gelesene Bytes=123
    Datei-Ausgabeformat-Zähler
        Geschriebene Bytes=23
```

Weiter schauen.

```
$ cat output/*
1   dfsadmin
```

Was bedeutet das nun? Keine Sorge, jedenfalls haben wir `Hadoop` zum Laufen gebracht. Und wir haben das erste Beispiel für eine lokale Berechnung ausgeführt.

Spark

Zurück zu Spark. Schauen wir uns ein Beispiel an.

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Hier ist eine `hdfs`-Datei aufgetaucht. Nach einiger Recherche habe ich herausgefunden, dass man eine `hdfs`-Datei auf folgende Weise erstellen kann:

```
hdfs dfs -mkdir /test
```

Schauen wir uns den `hdfs`-Befehl an.

```
$ hdfs
Verwendung: hdfs [OPTIONEN] UNTERBEFEHL [UNTERBEFEHLSOPTIONEN]
```

OPTIONS ist keine oder eine der folgenden:

–buildpaths versucht, Klassendateien aus dem Build-Verzeichnis hinzuzufügen –config dir Hadoop-Konfigurationsverzeichnis –daemon (start|status|stop) führt eine Aktion auf einem Daemon aus –debug aktiviert den Debug-Modus für Shell-Skripte –help zeigt Nutzungsinformationen an –hostnames list[,of,host,names] Hosts, die im Worker-Modus verwendet werden sollen –hosts filename Liste der Hosts, die im Worker-Modus verwendet werden sollen –loglevel level setzt den Log4j-Level für diesen Befehl –workers aktiviert den Worker-Modus

SUBCOMMAND ist eines der folgenden: Admin-Befehle:

cacheadmin Konfigurieren des HDFS-Caches crypto Konfigurieren von HDFS-Verschlüsselungszonen debug Ausführen eines Debug-Administrators zur Ausführung von HDFS-Debug-Befehlen dfsadmin Ausführen eines DFS-Administrator-Clients dfsrouteradmin Verwalten der Router-basierten Föderation ec Ausführen eines HDFS-ErasureCoding-CLI fsck Ausführen eines DFS-Dateisystem-Prüfprogramms haadmin Ausführen eines DFS-HA-Administrator-Clients jmxget Abrufen von JMX-exportierten Werten vom NameNode oder DataNode oev Anwenden des Offline-Edits-Viewers auf eine Edits-Datei oiv Anwenden des Offline-Fsimage-Viewers auf ein Fsimage oiv_legacy Anwenden des Offline-Fsimage-Viewers auf ein Legacy-Fsimage storagepolicies Auflisten/Abrufen/Setzen/Erfüllen von Block-Speicherrichtlinien

```
Client-Befehle:
```

classpath gibt den Klassenpfad aus, der benötigt wird, um das Hadoop-JAR und die erforderlichen Bibliotheken zu erhalten dfs führt einen Dateisystembefehl auf dem Dateisystem aus envvars zeigt die berechneten Hadoop-Umgebungsvariablen an fetchdt holt ein Delegation-Token vom NameNode getconf ruft Konfigurationswerte aus der Konfiguration ab groups zeigt die Gruppen an, zu denen Benutzer gehören lsSnapshottableDir listet alle vom aktuellen Benutzer besessenen Snapshottable-Verzeichnisse auf snapshotDiff vergleicht zwei Snapshots

eines Verzeichnisses oder vergleicht die aktuellen Verzeichnisinhalte mit einem Snapshot version gibt die Version aus

Daemon-Befehle:

balancer Führt ein Cluster-Balancing-Utility aus
datanode Führt einen DFS-Datanode aus
dfsrouter Führt den DFS-Router aus
diskbalancer Verteilt Daten gleichmäßig auf Festplatten eines bestimmten Knotens
httpfs Führt den HttpFS-Server aus, das HDFS-HTTP-Gateway
journalnode Führt den DFS-Journalnode aus
mover Führt ein Utility aus, um Blockreplikate über Speichertypen hinweg zu verschieben
namenode Führt den DFS-Namenode aus
nfs3 Führt ein NFS-Version-3-Gateway aus
portmap Führt einen Portmap-Dienst aus
secondarynamenode Führt den sekundären DFS-Namenode aus
sps Führt den externen StoragePolicySatisfier aus
zkfc Führt den ZK-Failover-Controller-Daemon aus

SUBCOMMAND kann Hilfe anzeigen, wenn es ohne Parameter oder mit -h aufgerufen wird.

□□□□□□□

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]")\
            .config('spark.driver.bindAddress', '127.0.0.1')\
            .getOrCreate()
sc = spark.sparkContext

text_file = sc.textFile("a.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
              .map(lambda word: (word, 1)) \
              .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("b.txt")
```

Es ist wichtig, `.config('spark.driver.bindAddress', '127.0.0.1')` zu beachten. Andernfalls wird der Fehler `Service 'sparkDriver' could not bind on a random free port. You may check whether configuring an appropriate binding address` auftreten.

Allerdings trat zu diesem Zeitpunkt erneut ein Fehler auf.

```
Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/pyspark/python/lib/pyspark.zip/pyspark/worker.py", line 4
    raise Exception(("Python in worker has different version %s than that in " +
Exception: Python im Worker hat eine andere Version 3.8 als im Driver 3.9. PySpark kann nicht mit unters
```

zeigt an, dass verschiedene Versionen von `Python` ausgeführt wurden.

Ändern der `.bash_profile`:

```
PYSPARK_PYTHON=/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3
PYSPARK_DRIVER_PYTHON=/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3
```

Es wird jedoch weiterhin der gleiche Fehler gemeldet. Nach einiger Recherche könnte der Grund dafür sein, dass `spark` beim Ausführen diese Umgebungsvariable nicht geladen hat und nicht die Standard-Umgebungsvariablen des Terminals verwendet.

In der Code müssen Sie folgendes einstellen:

```
import os
```

Spark-Umgebungen setzen

os.environ[‘PYSPARK_PYTHON’] = ‘/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3’ os.environ[‘PYSPARK_DR
= ‘/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3’

```
Das wird ausgeführt.


```shell
$ python sc.py
21/03/11 02:54:52 WARN NativeCodeLoader: Native-Hadoop-Bibliothek konnte für Ihre Plattform nicht gelad
Verwende Sparks Standard-Log4j-Profil: org/apache/spark/log4j-defaults.properties
Standard-Log-Level auf "WARN" gesetzt.
Um das Logging-Level anzupassen, verwenden Sie sc.setLogLevel(newLevel). Für SparkR verwenden Sie setLo
PythonRDD[6] at RDD at PythonRDD.scala:53
```
```

Zu diesem Zeitpunkt wurde `b.txt` erstellt.

```
  b.txt
      _SUCCESS
      part-00000
      part-00001
```

Öffne es.

```
$ cat b.txt/part-00000
('college', 1)
('two', 1)
('things', 2)
('worked', 1)
('on,', 1)
('of', 8)
('school,', 2)
('writing', 2)
('programming.', 1)
("didn't", 4)
('then,', 1)
('probably', 1)
('are:', 1)
('short', 1)
('awful.', 1)
('They', 1)
('plot,', 1)
('just', 1)
('characters', 1)
('them', 2)
...
```

Erfolg! Kommt dir das bekannt vor? Es ist genau wie im `Hadoop`-Beispiel.

```
$ cat output/*
1   dfsadmin
```

Diese Dateien werden `HDFS` genannt. Hier wird `Spark` verwendet, um Wörter zu zählen. Mit nur wenigen Zeilen sieht es sehr praktisch aus.

Kubernetes

Als nächstes beschäftige ich mich mit `Kubernetes`, auch bekannt als `k8s`, wobei die 8 die acht ausgelassenen Buchstaben in der Mitte darstellt. Es handelt sich um ein Open-Source-System, das die Automatisierung der Bereitstellung, Skalierung und Verwaltung von Containeranwendungen ermöglicht.

Das `kubectl`-Befehlszeilentool wird verwendet, um Befehle auf einem Kubernetes-Cluster auszuführen. Es kann verwendet werden, um Anwendungen bereitzustellen, Cluster-Ressourcen anzuzeigen und zu verwalten sowie Protokolle einzusehen.

Man kann es auch mit Homebrew installieren.

```
brew install kubectl
```

Protokollausgabe:

```
==> Herunterladen von https://homebrew.bintray.com/bottles/kubernetes-cli-1.20.1.big_sur.bottle.tar.gz
==> Herunterladen von https://d29vzk4ow07wi7.cloudfront.net/0b4f08bd1d47cb913d7cd4571e3394c6747dfbad7ff
######################################################################## 100.0%
==> Entpacken von kubernetes-cli-1.20.1.big_sur.bottle.tar.gz
==> Hinweise
Die Bash-Vervollständigung wurde installiert in:
  /usr/local/etc/bash_completion.d
==> Zusammenfassung
   /usr/local/Cellar/kubernetes-cli/1.20.1: 246 Dateien, 46,1MB
```

Installation abgeschlossen.

```
$ kubectl version --client
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
```

```
$ kubectl
kubectl steuert den Kubernetes-Cluster-Manager.
```

Weitere Informationen finden Sie unter: https://kubernetes.io/docs/reference/kubectl/overview/

Grundlegende Befehle (Anfänger): create Erstellt eine Ressource aus einer Datei oder von stdin. expose Nimmt einen Replication Controller, Service, Deployment oder Pod und macht

ihn als neuen Kubernetes Service verfügbar. run Führt ein bestimmtes Image im Cluster aus. set Legt spezifische Funktionen für Objekte fest.

Grundlegende Befehle (Fortgeschrittene): explain Dokumentation von Ressourcen get Eine oder mehrere Ressourcen anzeigen edit Eine Ressource auf dem Server bearbeiten delete Ressourcen löschen anhand von Dateinamen, stdin, Ressourcen und Namen oder durch Ressourcen und Label-Selektor

Befehle für die Bereitstellung: rollout Verwaltet das Rollout einer Ressource scale Legt eine neue Größe für eine Deployment, ReplicaSet oder Replication Controller fest autoscale Skaliert automatisch ein Deployment, ReplicaSet oder ReplicationController

Cluster Management Befehle: certificate Zertifikatsressourcen ändern. cluster-info Cluster-Informationen anzeigen top Ressourcenverbrauch (CPU/Speicher/Speicherplatz) anzeigen. cordon Knoten als nicht planbar markieren uncordon Knoten als planbar markieren drain Knoten für Wartungsarbeiten vorbereiten taint Taints auf einem oder mehreren Knoten aktualisieren

Fehlerbehebung und Debugging-Befehle: describe Zeigt Details einer bestimmten Ressource oder einer Gruppe von Ressourcen an logs Gibt die Protokolle eines Containers in einem Pod aus attach Verbindet sich mit einem laufenden Container exec Führt einen Befehl in einem Container aus port-forward Leitet einen oder mehrere lokale Ports an einen Pod weiter proxy Startet einen Proxy zum Kubernetes API-Server cp Kopiert Dateien und Verzeichnisse zu und von Containern auth Überprüft die Autorisierung debug Erstellt Debugging-Sitzungen zur Fehlerbehebung von Workloads und Knoten

Erweiterte Befehle: diff Vergleicht die Live-Version mit der Version, die angewendet werden würde apply Wendet eine Konfiguration auf eine Ressource an, basierend auf einem Dateinamen oder stdin patch Aktualisiert Feld(er) einer Ressource replace Ersetzt eine Ressource basierend auf einem Dateinamen oder stdin wait Experimentell: Wartet auf eine bestimmte Bedingung für eine oder mehrere Ressourcen. kustomize Erstellt ein Kustomization-Ziel aus einem Verzeichnis oder einer Remote-URL.

Einstellungsbefehle: label Aktualisiert die Labels einer Ressource annotate Aktualisiert die Annotationen einer Ressource completion Gibt den Shell-Vervollständigungscode für die angegebene Shell aus (bash oder zsh)

Weitere Befehle: api-resources Zeigt die unterstützten API-Ressourcen auf dem Server an api-versions Zeigt die unterstützten API-Versionen auf dem Server in der Form “Gruppe/Version” an config Bearbeitet kubeconfig-Dateien plugin Bietet Hilfsmittel für die Interaktion mit Plugins. version Zeigt die Versionsinformationen des Clients und des Servers an

Verwendung: kubectl [flags] [options]

Verwenden Sie “kubectl –help”, um weitere Informationen zu einem bestimmten Befehl zu erhalten. Verwenden Sie “kubectl options”, um eine Liste globaler Befehlszeilenoptionen anzuzeigen (gilt für alle Befehle).

Erstellen wir eine Konfigurationsdatei.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Der obige Codeblock enthält keine spezifischen Anweisungen oder Inhalte, die übersetzt werden müssen. Es

```shell
$ kubectl apply -f simple_deployment.yaml
Die Verbindung zum Server localhost:8080 wurde abgelehnt - haben Sie den richtigen Host oder Port angeg

$ kubectl cluster-info
```

Um Cluster-Probleme weiter zu debuggen und zu diagnostizieren, verwenden Sie `kubectl`

`cluster-info dump`. Die Verbindung zum Server localhost:8080 wurde abgelehnt – haben Sie den richtigen Host oder Port angegeben?

Wenn Sie es versuchen, im Terminal der offiziellen Website auszuführen.

```shell
$ start.sh
Starte Kubernetes...minikube Version: v1.8.1
Commit: cbda04cf6bbe65e987ae52bb393c10099ab62014
* minikube v1.8.1 auf Ubuntu 18.04
* Verwendung des none-Treibers basierend auf Benutzerkonfiguration
* Läuft auf localhost (CPUs=2, Speicher=2460MB, Festplatte=145651MB) ...
* Betriebssystemversion ist Ubuntu 18.04.4 LTS
```

- Vorbereitung von Kubernetes v1.17.3 auf Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
- Starten von Kubernetes ...
- Aktivieren von Addons: default-storageclass, storage-provisioner
- Konfiguration der lokalen Host-Umgebung ...
- Fertig! kubectl ist nun so konfiguriert, dass es “minikube” verwendet
- Das ‘dashboard’-Addon ist aktiviert Kubernetes gestartet

Kehren wir zurück zu unserem Terminal.

```shell
$ kubectl version --client
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.1", GitCommit:"c4d752765b3bbac223
Die Verbindung zum Server localhost:8080 wurde abgelehnt - haben Sie den richtigen Host oder Port angeg
```

Interessanterweise führt das Hinzufügen der Option `--client` nicht zu einem Fehler.

Die Dokumentation besagt, dass zuerst `Minikube` installiert werden muss.

```
$ brew install minikube
==> Lade https://homebrew.bintray.com/bottles/minikube-1.16.0.big_sur.bottle.tar.gz herunter
```

```
==> Lade von https://d29vzk4ow07wi7.cloudfront.net/1b6d7d1b97b11b6b07e4fa531c2dc21770da290da9b2816f360f
######################################################################## 100.0%
==> Gieße minikube-1.16.0.big_sur.bottle.tar.gz ein
==> Hinweise
Die Bash-Vervollständigung wurde installiert in:
  /usr/local/etc/bash_completion.d
==> Zusammenfassung
   /usr/local/Cellar/minikube/1.16.0: 8 Dateien, 64.6MB

$ minikube start
   minikube v1.16.0 auf Darwin 11.2.2
   minikube 1.18.1 ist verfügbar! Lade es herunter: https://github.com/kubernetes/minikube/releases/tag/
   Um diese Benachrichtigung zu deaktivieren, führe aus: 'minikube config set WantUpdateNotification fal
```

 Automatisch den Virtualbox-Treiber ausgewählt VM-Boot-Image wird heruntergeladen ... > minikube-v1.16.0.iso.sha256: 65 B / 65 B [————-] 100.00% ? p/s 0s > minikube-v1.16.0.iso: 212.62 MiB / 212.62 MiB [] 100.00% 5.32 MiB p/s 40s Starte den Control-Plane-Knoten minikube im Cluster minikube Kubernetes v1.20.0 Preload wird heruntergeladen ... > preloaded-images-k8s-v8-v1....: 491.00 MiB / 491.00 MiB 100.00% 7.52 MiB Erstelle Virtualbox-VM (CPUs=2, Speicher=4000MB, Festplatte=20000MB) ... Diese VM hat Probleme, auf https://k8s.gcr.io zuzugreifen Um neue externe Images zu pullen, müssen Sie möglicherweise einen Proxy konfigurieren: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/ Bereite Kubernetes v1.20.0 auf Docker 20.10.0 vor ... ▪ Zertifikate und Schlüssel werden generiert ... ▪ Control Plane wird hochgefahren ... ▪ RBAC-Regeln werden konfiguriert ... Überprüfe Kubernetes-Komponenten... Aktivierte Addons: storage-provisioner, default-storageclass Fertig! kubectl ist nun standardmäßig für die Verwendung des Clusters “minikube” und des Namespace “default” konfiguriert

```
Als Nächstes greifen wir auf diesen Cluster zu.

```shell
$ kubectl get po -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-74ff55c5b-ndbcr            1/1     Running   0          60s
kube-system   etcd-minikube                      0/1     Running   0          74s
kube-system   kube-apiserver-minikube            1/1     Running   0          74s
kube-system   kube-controller-manager-minikube   1/1     Running   0          74s
```
```

```
kube-system   kube-proxy-g2296                1/1     Running   0          60s
kube-system   kube-scheduler-minikube         0/1     Running   0          74s
kube-system   storage-provisioner             1/1     Running   1          74s
```

Um das Dashboard von `minikube` zu öffnen, führen Sie den folgenden Befehl aus:

```
minikube dashboard
```

Dieser Befehl startet das Kubernetes-Dashboard und öffnet es in Ihrem Standard-Webbrowser.

```
$ minikube dashboard
   Dashboard wird aktiviert ...
   Überprüfung der Dashboard-Integrität ...
   Proxy wird gestartet ...
   Überprüfung der Proxy-Integrität ...
   Öffnen von http://127.0.0.1:50030/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-das
```

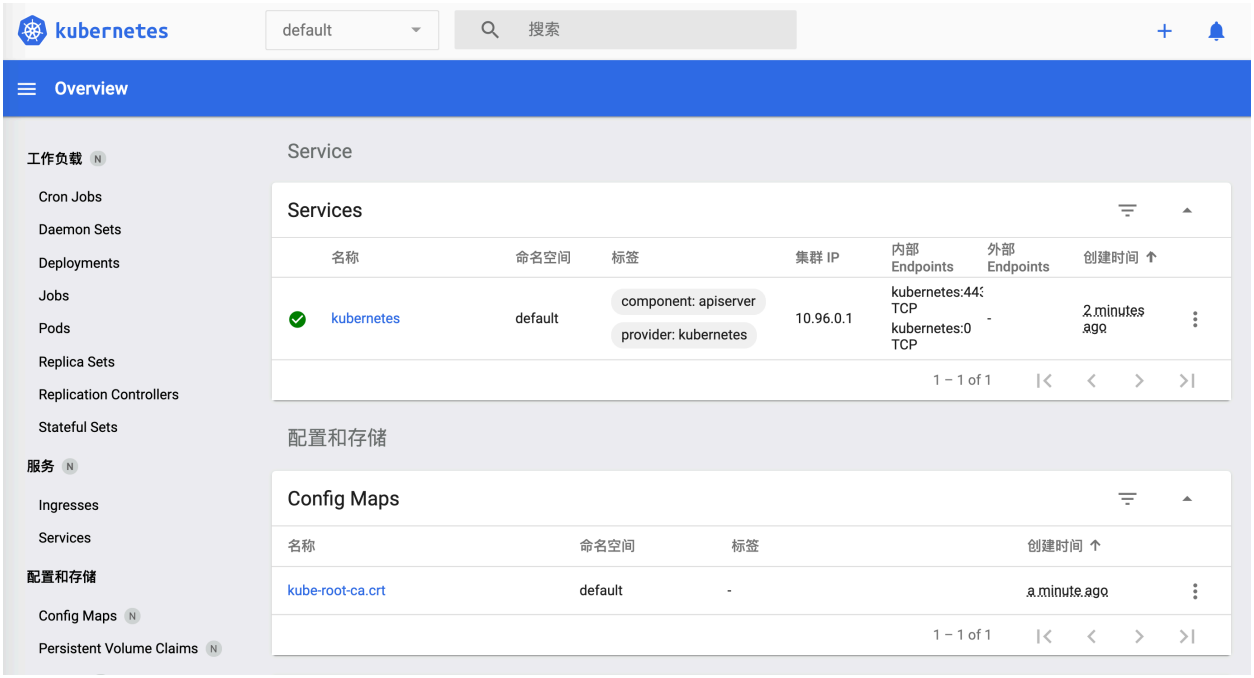

Figure 1: k8s

Wie schaltet man es aus?

```
$ minikube
minikube stellt lokale Kubernetes-Cluster bereit und verwaltet sie, die für Entwicklungs-Workflows optim
```

Grundlegende Befehle: start Startet einen lokalen Kubernetes-Cluster status Ruft den Status eines lokalen Kubernetes-Clusters ab stop Stoppt einen laufenden lokalen Kubernetes-Cluster delete Löscht einen lokalen Kubernetes-Cluster dashboard Greift auf das Kubernetes-Dashboard zu, das im Minikube-Cluster läuft pause Pausiert Kubernetes unpause Setzt Kubernetes fort

Bilder-Befehle: docker-env Konfiguriert die Umgebung, um den Docker-Daemon von minikube zu verwenden podman-env Konfiguriert die Umgebung, um den Podman-Dienst von minikube zu verwenden cache Fügt ein lokales Bild hinzu, löscht es oder lädt es in minikube hoch

Konfigurations- und Verwaltungsbefehle: addons Aktivieren oder deaktivieren Sie ein Minikube-Addon config Ändern Sie persistente Konfigurationswerte profile Abrufen oder Auflisten der aktuellen Profile (Cluster) update-context Aktualisieren Sie kubeconfig im Falle einer IP- oder Portänderung

Netzwerk- und Konnektivitätsbefehle: service Gibt eine URL zurück, um eine Verbindung zu einem Dienst herzustellen tunnel Verbindung zu LoadBalancer-Diensten herstellen

Erweiterte Befehle: mount Bindet das angegebene Verzeichnis in minikube ein ssh Loggt sich in die minikube-Umgebung ein (für Debugging-Zwecke) kubectl Führt eine kubectl-Binärdatei aus, die der Cluster-Version entspricht node Fügt zusätzliche Knoten hinzu, entfernt sie oder listet sie auf

Fehlerbehebungsbefehle: ssh-key Ruft den Pfad des SSH-Identitätsschlüssels des angegebenen Knotens ab ssh-host Ruft den SSH-Hostschlüssel des angegebenen Knotens ab ip Ruft die IP-Adresse des angegebenen Knotens ab logs Gibt Protokolle zurück, um einen lokalen Kubernetes-Cluster zu debuggen update-check Druckt die aktuelle und die neueste Versionsnummer version Druckt die Version von minikube

Andere Befehle: completion Generiert Befehlsvervollständigung für eine Shell

Verwenden Sie `minikube <Befehl> --help`, um weitere Informationen zu einem bestimmten Befehl zu erhalten.

```
Es scheint, dass es sich um `minikube stop` handelt.

Zurück zu `kubernetes`, jetzt funktioniert alles einwandfrei.

```shell
$ kubectl cluster-info
Kubernetes Control Plane läuft unter https://192.168.99.100:8443
```
```

KubeDNS läuft unter https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/pro

Um Cluster-Probleme weiter zu debuggen und zu diagnostizieren, verwenden Sie `kubectl cluster-info dump`.

Wenn wir `https://192.168.99.100:8443` öffnen, zeigt der Browser:

```json
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "verboten: Benutzer \"system:anonymous\" kann den Pfad \"/\" nicht abrufen",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}
```

□□ https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy□


```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Fehler",
  "message": "Dienste \"kube-dns:dns\" sind verboten: Benutzer \"system:anonymous\" kann die Ressource
  "reason": "Verboten",
  "details": {
    "name": "kube-dns:dns",
    "kind": "services"
```

```
  },
  "code": 403
}
```

Probieren wir die gerade besprochene Konfiguration aus.

```
$ kubectl apply -f simple_deployment.yaml
deployment.apps/nginx-deployment erstellt
```

Es gab ein kleines Problem. Aber bis hierher haben wir `Kubernetes` zum Laufen gebracht. Beenden wir es erstmal. Wir werden später weiter damit spielen.

```
$ minikube stop
  Stoppe den Knoten "minikube"  ...
  1 Knoten gestoppt.
```

Überprüfen, ob beendet ist.

```
w$ minikube dashboard
  Der Control-Plane-Knoten muss ausgeführt werden, um diesen Befehl auszuführen
  Um einen Cluster zu starten, führen Sie aus: "minikube start"
```

Docker

`Docker` ist ebenfalls eine Container-Plattform, die dazu beiträgt, die Erstellung, Freigabe und Ausführung moderner Anwendungen zu beschleunigen. Laden Sie die Anwendung von der offiziellen Website herunter.

Die Verwendung des Clients ist etwas langsam. Lass uns die Befehlszeile verwenden.

```
$ docker
```

Verwendung: docker [OPTIONEN] BEFEHL

Eine eigenständige Laufzeitumgebung für Container

Optionen: –config string Speicherort der Client-Konfigurationsdateien (Standard “/Users/lzw/.docker”) -c, –context string Name des Kontexts, der für die Verbindung zum Daemon verwendet werden soll (überschreibt die DOCKER_HOST-Umgebungsvariable und den Standardkontext, der

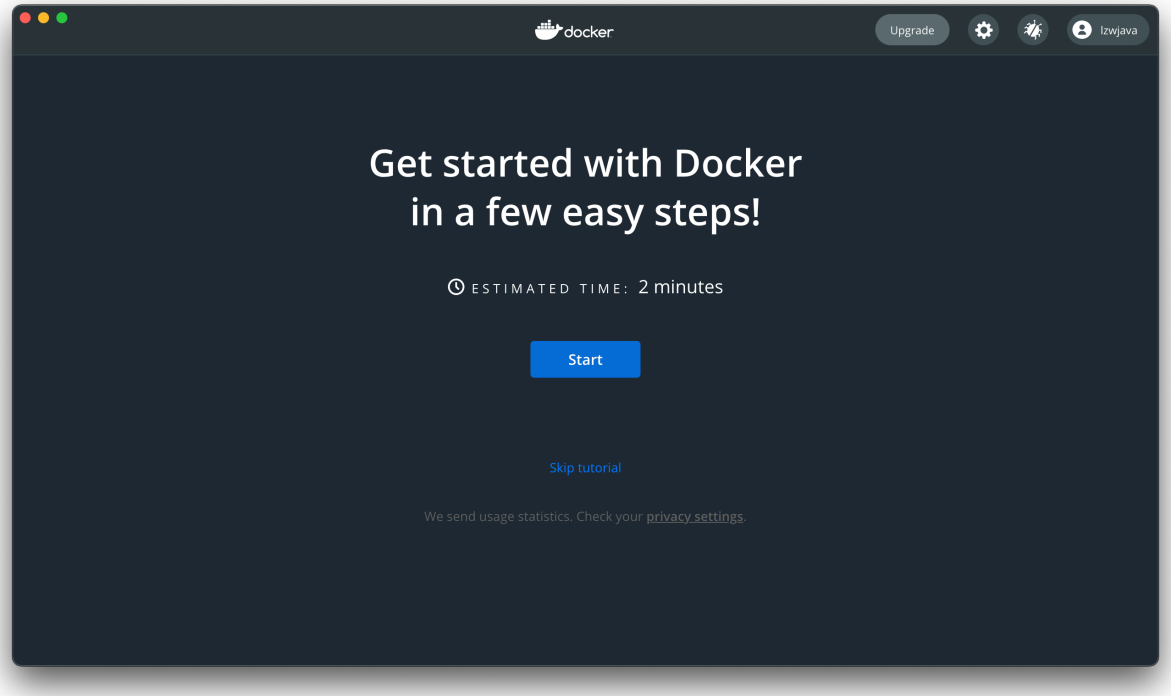

Figure 2: docker

mit “docker context use” festgelegt wurde) -D, –debug Debug-Modus aktivieren -H, –host list Daemon-Socket(s), zu dem/denen eine Verbindung hergestellt werden soll -l, –log-level string Legt das Log-Level fest (“debug”|“info”|“warn”|“error”|“fatal”) (Standard “info”) –tls TLS verwenden; impliziert durch –tlsverify –tlscacert string Nur Zertifikate vertrauen, die von dieser CA signiert wurden (Standard “/Users/lzw/.docker/ca.pem”) –tlscert string Pfad zur TLS-Zertifikatsdatei (Standard “/Users/lzw/.docker/cert.pem”) –tlskey string Pfad zur TLS-Schlüsseldatei (Standard “/Users/lzw/.docker/key.pem”) –tlsverify TLS verwenden und die Remote-Verbindung überprüfen -v, –version Versionsinformationen anzeigen und beenden

Management-Befehle: app* Docker App (Docker Inc., v0.9.1-beta3) builder Builds verwalten buildx* Mit BuildKit bauen (Docker Inc., v0.5.1-docker) config Docker-Konfigurationen verwalten container Container verwalten context Kontexte verwalten image Images verwalten manifest Docker-Image-Manifeste und Manifest-Listen verwalten network Netzwerke verwalten node Swarm-Knoten verwalten plugin Plugins verwalten scan* Docker Scan (Docker Inc., v0.5.0) secret Docker-Geheimnisse verwalten service Dienste verwalten stack Docker-Stacks verwalten swarm Swarm verwalten system Docker verwalten trust Vertrauenswürdigkeit von Docker-Images verwalten volume Volumes verwalten

Befehle: attach Lokale Standard-Eingabe-, Ausgabe- und Fehlerströme an einen laufenden

Container anhängen build Ein Image aus einer Dockerfile erstellen commit Ein neues Image aus den Änderungen eines Containers erstellen cp Dateien/Ordner zwischen einem Container und dem lokalen Dateisystem kopieren create Einen neuen Container erstellen diff Änderungen an Dateien oder Verzeichnissen im Dateisystem eines Containers überprüfen events Echtzeit-Ereignisse vom Server abrufen exec Einen Befehl in einem laufenden Container ausführen export Das Dateisystem eines Containers als tar-Archiv exportieren history Die Historie eines Images anzeigen images Images auflisten import Den Inhalt eines tar-Archivs importieren, um ein Dateisystem-Image zu erstellen info Systemweite Informationen anzeigen inspect Niedrigstufige Informationen zu Docker-Objekten zurückgeben kill Einen oder mehrere laufende Container beenden load Ein Image aus einem tar-Archiv oder STDIN laden login Bei einer Docker-Registry anmelden logout Von einer Docker-Registry abmelden logs Die Protokolle eines Containers abrufen pause Alle Prozesse in einem oder mehreren Containern anhalten port Port-Zuordnungen oder eine spezifische Zuordnung für den Container auflisten ps Container auflisten pull Ein Image oder ein Repository aus einer Registry herunterladen push Ein Image oder ein Repository in eine Registry hochladen rename Einen Container umbenennen restart Einen oder mehrere Container neu starten rm Einen oder mehrere Container entfernen rmi Einen oder mehrere Images entfernen run Einen Befehl in einem neuen Container ausführen save Ein oder mehrere Images in ein tar-Archiv speichern (standardmäßig nach STDOUT gestreamt) search Im Docker Hub nach Images suchen start Einen oder mehrere gestoppte Container starten stats Einen Live-Stream der Ressourcennutzungsstatistiken von Container(n) anzeigen stop Einen oder mehrere laufende Container stoppen tag Ein Tag TARGET_IMAGE erstellen, das auf SOURCE_IMAGE verweist top Die laufenden Prozesse eines Containers anzeigen unpause Alle Prozesse in einem oder mehreren Containern fortsetzen update Die Konfiguration eines oder mehrerer Container aktualisieren version Die Docker-Versionsinformationen anzeigen wait Blockieren, bis ein oder mehrere Container stoppen, und dann deren Exit-Codes ausgeben

Führen Sie `docker COMMAND --help` aus, um weitere Informationen zu einem Befehl zu erhalten.

Um mehr Hilfe zu Docker zu erhalten, schauen Sie sich unsere Anleitungen unter https://docs.docker.com/go/gu
an.

```
Folgen Sie dem Tutorial und probieren Sie es aus.

```shell
$ docker run -d -p 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
aad63a933944: Pull complete
```
```

```
b14da7a62044: Pull complete
343784d40d66: Pull complete
6f617e610986: Pull complete
Digest: sha256:d2c4fb0641519ea208f20ab03dc40ec2a5a53fdfbccca90bef14f870158ed577
Status: Downloaded newer image for docker/getting-started:latest
815f13fc8f99f6185257980f74c349e182842ca572fe60ff62cbb15641199eaf
docker: Error response from daemon: Ports are not available: listen tcp 0.0.0.0:80: bind: address alrea
```

Übersetzung:

```
$ docker run -d -p 80:80 docker/getting-started
Das Image 'docker/getting-started:latest' wurde lokal nicht gefunden.
latest: Wird von docker/getting-started gezogen
aad63a933944: Pull complete
b14da7a62044: Pull complete
343784d40d66: Pull complete
6f617e610986: Pull complete
Digest: sha256:d2c4fb0641519ea208f20ab03dc40ec2a5a53fdfbccca90bef14f870158ed577
Status: Neueres Image für docker/getting-started wurde heruntergeladen.
815f13fc8f99f6185257980f74c349e182842ca572fe60ff62cbb15641199eaf
docker: Fehlerantwort vom Daemon: Ports sind nicht verfügbar: listen tcp 0.0.0.0:80: bind: Adresse wird
```

Ändere den Port.

```
$ docker run -d -p 8080:80 docker/getting-started
45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
```

Öffnen Sie den Browser, um zu zeigen, dass wir `docker` erfolgreich gestartet haben.

Stoppe den Container. Verwende die zuvor zurückgegebene `ID`.

```
$ docker stop 45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
45bb95fa1ae80adc05cc498a1f4f339c45c51f7a8ae1be17f5b704853a5513a5
```

Zu diesem Zeitpunkt war die Website bereits nicht mehr erreichbar.

Das deutet darauf hin, dass `docker` wie eine virtuelle Maschine funktioniert.

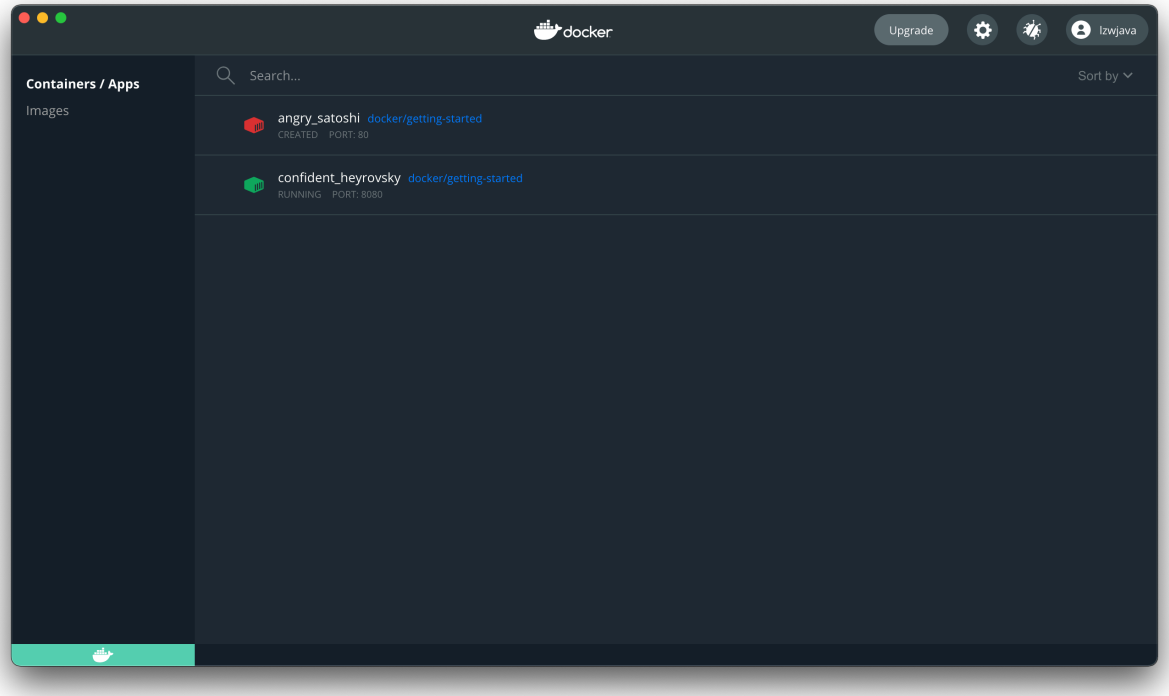

Figure 3: docker_run

Flink

Öffnen Sie die offizielle Website.

`Flink` spricht von `Stateful`-Berechnungen für Datenströme. Was bedeutet `Stateful`? Das ist mir noch nicht ganz klar. Das obige Diagramm ist jedoch sehr interessant. Lass es uns ausprobieren.

Es wird eine Java-Umgebung benötigt.

```
$ java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (Build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (Build 25.151-b12, gemischter Modus)
```

Laden Sie die neueste Version `flink-1.12.2-bin-scala_2.11.tar` von der offiziellen Website herunter.

```
$ ./bin/start-cluster.sh
Cluster wird gestartet.
```

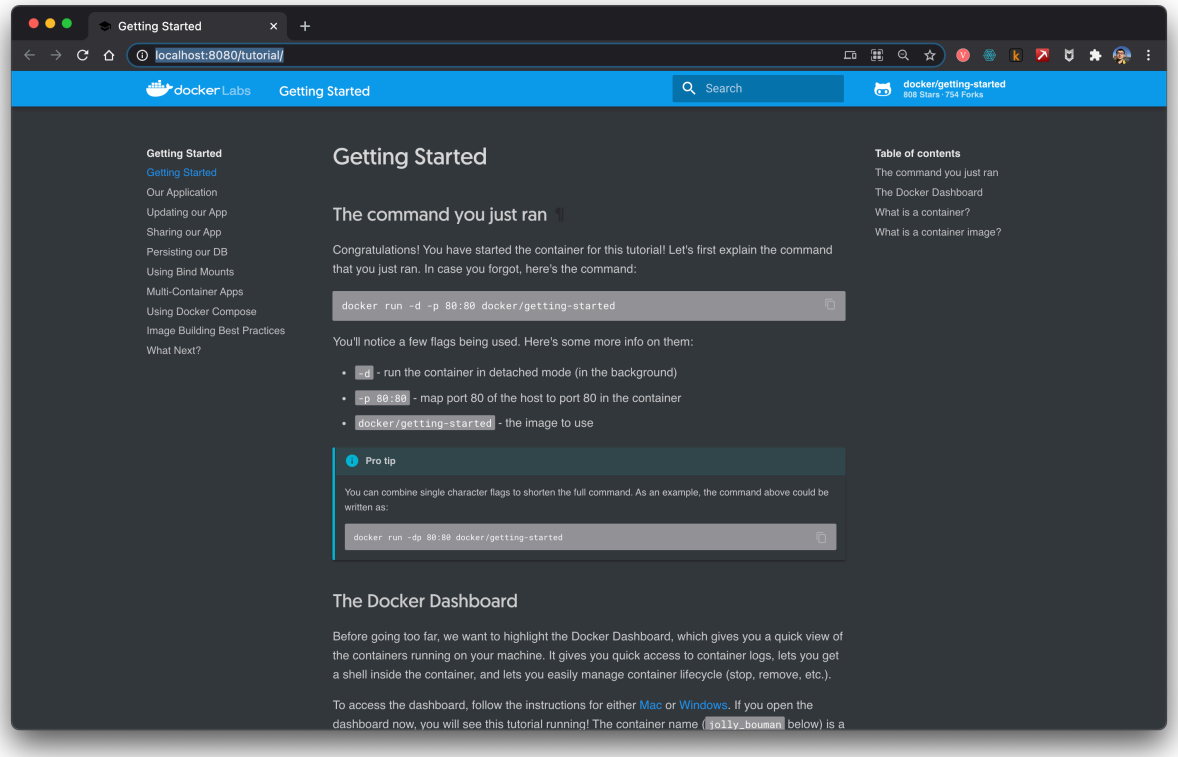

Figure 4: Browser

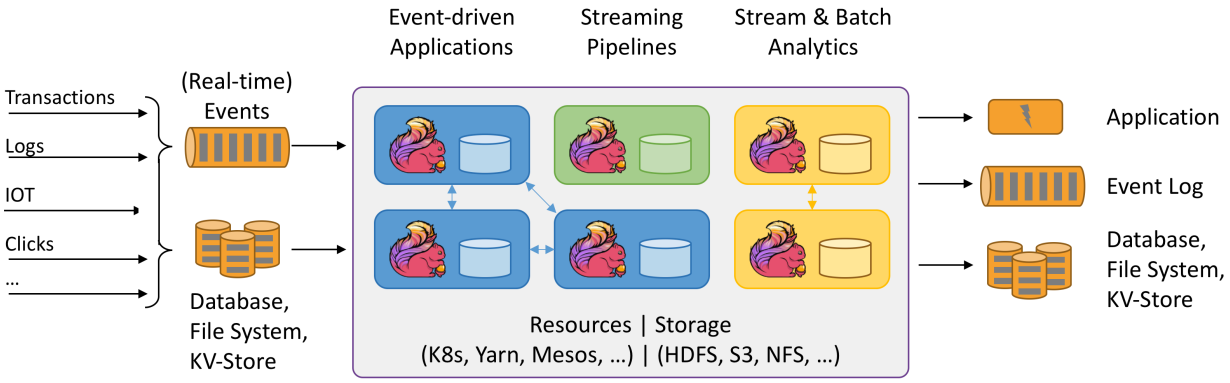

Figure 5: flink-home-graphic

```
Standalonesession-Daemon wird auf Host lzwjava gestartet.
Taskexecutor-Daemon wird auf Host lzwjava gestartet.

$ ./bin/flink run examples/streaming/WordCount.jar
Ausführung des WordCount-Beispiels mit dem Standard-Eingabedatensatz.
Verwenden Sie --input, um eine Dateieingabe anzugeben.
Das Ergebnis wird auf stdout ausgegeben. Verwenden Sie --output, um einen Ausgabepfad anzugeben.
Der Job wurde mit der JobID 60f37647c20c2a6654359bd34edab807 übermittelt.
Programmausführung abgeschlossen
Der Job mit der JobID 60f37647c20c2a6654359bd34edab807 ist abgeschlossen.
Job-Laufzeit: 757 ms

$ tail log/flink-*-taskexecutor-*.out
(nymph,1)
(in,3)
(thy,1)
(orisons,1)
(be,4)
(all,2)
(my,1)
(sins,1)
(remember,1)
(d,4)

$ ./bin/stop-cluster.sh
Taskexecutor-Daemon wird gestoppt (pid: 41812) auf Host lzwjava.
```

Ja, der Einstieg war erfolgreich. Man kann sehen, dass dies `Spark` sehr ähnlich ist.

Kylin

Öffnen Sie die offizielle Website.

> Apache Kylin™ ist ein Open-Source, verteiltes Analytisches Data Warehouse für Big Data; es wurde entwickelt, um OLAP-Fähigkeiten (Online Analytical Processing) im Zeitalter von Big Data bereitzustellen. Durch die Modernisierung der Multi-Dimensional-Cube- und Vorberechnungstechnologie auf Hadoop und Spark ist Kylin

in der Lage, nahezu konstante Abfragegeschwindigkeiten zu erreichen, unabhängig vom stetig wachsenden Datenvolumen. Indem Kylin die Abfragelatenz von Minuten auf unter eine Sekunde reduziert, bringt es Online-Analysen zurück zu Big Data.

Apache Kylin™ ermöglicht es Ihnen, Milliarden von Zeilen in weniger als einer Sekunde in 3 Schritten abzufragen.

1. Identifizieren Sie ein Stern- oder Schneeflockenschema auf Hadoop.
2. Erstellen Sie einen Cube aus den identifizierten Tabellen.
3. Führen Sie Abfragen mit ANSI-SQL durch und erhalten Sie Ergebnisse in weniger als einer Sekunde über ODBC, JDBC oder eine RESTful API.

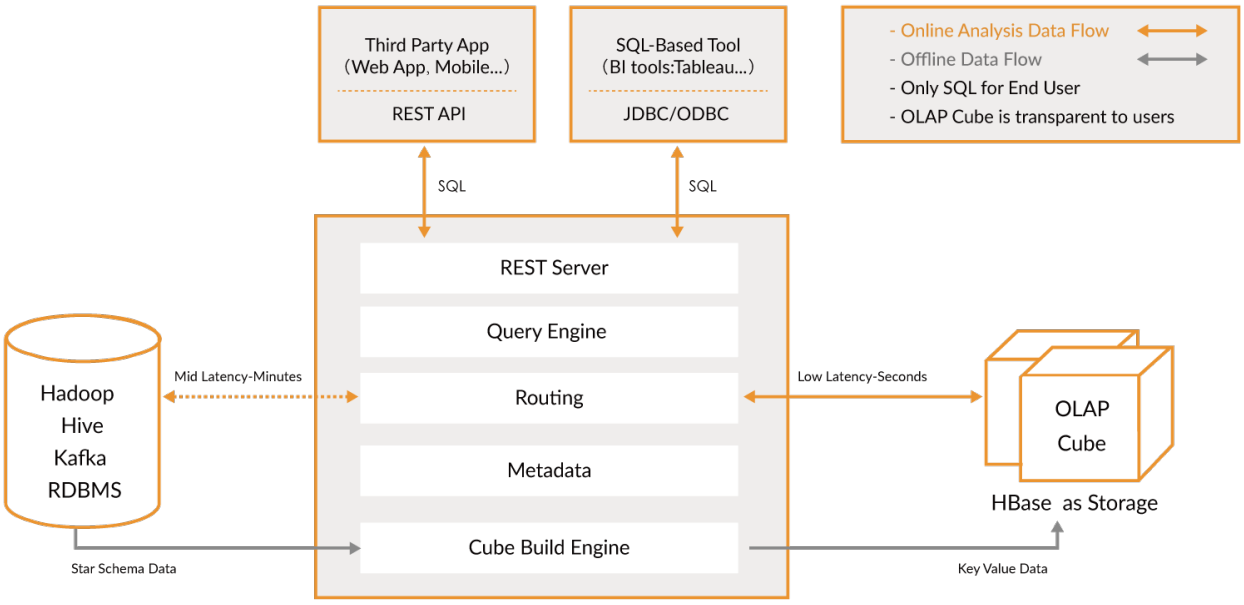

Figure 6: kylin_diagram

Es handelt sich im Wesentlichen um eine Ebene zur Analyse von Big Data. Mit ihr kann man sehr schnell suchen. Sie dient als Brücke.

Leider ist die Nutzung derzeit nur in einer `Linux`-Umgebung möglich. Ich werde später noch einmal daran herumspielen.

MongoDB

Das ist auch eine Art von Datenbank. Versuchen Sie, es zu installieren.

```
$ brew tap mongodb/brew
```

```
==> Tapping mongodb/brew
Klone nach '/usr/local/Homebrew/Library/Taps/mongodb/homebrew-brew'...
remote: Enumerating objects: 63, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 566 (delta 21), reused 6 (delta 1), pack-reused 503
Empfange Objekte: 100% (566/566), 121.78 KiB | 335.00 KiB/s, done.
Löse Deltas auf: 100% (259/259), done.
11 Formeln angezapft (39 Dateien, 196.2KB).

$ brew install mongodb-community@4.4
==> Installation von mongodb-community aus mongodb/brew
==> Herunterladen von https://fastdl.mongodb.org/tools/db/mongodb-database-tools-macos-x86_64-100.3.0.z
######################################################################## 100.0%
==> Herunterladen von https://fastdl.mongodb.org/osx/mongodb-macos-x86_64-4.4.3.tgz
######################################################################## 100.0%
==> Installation der Abhängigkeiten für mongodb/brew/mongodb-community: mongodb-database-tools
==> Installation der Abhängigkeit mongodb/brew/mongodb-community: mongodb-database-tools
Fehler: Der `brew link`-Schritt wurde nicht erfolgreich abgeschlossen
Die Formel wurde gebaut, ist aber nicht in /usr/local verlinkt
Konnte bin/bsondump nicht verlinken
Ziel /usr/local/bin/bsondump
ist ein Symlink, der zu mongodb gehört. Sie können ihn entlinken:
  brew unlink mongodb
```

Um den Link zu erzwingen und alle konfligierenden Dateien zu überschreiben: brew link –overwrite mongodb-database-tools

Um alle Dateien aufzulisten, die gelöscht würden: `bash brew link --overwrite --dry-run mongodb-database-tools`

Mögliche konfliktierende Dateien sind: /usr/local/bin/bsondump -> /usr/local/Cellar/mongodb/3.0.7/bin/bsondu /usr/local/bin/mongodump -> /usr/local/Cellar/mongodb/3.0.7/bin/mongodump /usr/local/bin/mongoexport -> /usr/local/Cellar/mongodb/3.0.7/bin/mongoexport /usr/local/bin/mongofiles -> /usr/local/Cellar/mongodb/3 /usr/local/bin/mongoimport -> /usr/local/Cellar/mongodb/3.0.7/bin/mongoimport /usr/local/bin/mongorestore -> /usr/local/Cellar/mongodb/3.0.7/bin/mongorestore /usr/local/bin/mongostat -> /usr/local/Cellar/mongodb/3 /usr/local/bin/mongotop -> /usr/local/Cellar/mongodb/3.0.7/bin/mongotop ==> Zusammenfassung /usr/local/Cellar/mongodb-database-tools/100.3.0: 13 Dateien, 154MB, in 11

Sekunden erstellt ==> Installation von mongodb/brew/mongodb-community Fehler: Der `brew link` Schritt wurde nicht erfolgreich abgeschlossen Die Formel wurde erstellt, ist jedoch nicht in /usr/local verlinkt Konnte bin/mongo nicht verlinken Ziel /usr/local/bin/mongo ist ein Symlink, der zu mongodb gehört. Sie können ihn entlinken: brew unlink mongodb

Um den Link zu erzwingen und alle konfligierenden Dateien zu überschreiben: brew link –overwrite mongodb-community

Um alle Dateien aufzulisten, die gelöscht würden: brew link –overwrite –dry-run mongodb-community

Mögliche konfliktierende Dateien sind: /usr/local/bin/mongo -> /usr/local/Cellar/mongodb/3.0.7/bin/mongo /usr/local/bin/mongod -> /usr/local/Cellar/mongodb/3.0.7/bin/mongod /usr/local/bin/mongos -> /usr/local/Cellar/mongodb/3.0.7/bin/mongos ==> Hinweise Um launchd zu veranlassen, mongodb/brew/mongodb-community jetzt zu starten und bei der Anmeldung neu zu starten: brew services start mongodb/brew/mongodb-community Oder, wenn Sie keinen Hintergrunddienst benötigen/wollen, können Sie einfach ausführen: mongod –config /usr/local/etc/mongod.conf ==> Zusammenfassung /usr/local/Cellar/mongodb-community/4.4.3: 11 Dateien, 156,8 MB, in 10 Sekunden erstellt ==> Hinweise ==> mongodb-community Um launchd zu veranlassen, mongodb/brew/mongodb-community jetzt zu starten und bei der Anmeldung neu zu starten: brew services start mongodb/brew/mongodb-community Oder, wenn Sie keinen Hintergrunddienst benötigen/wollen, können Sie einfach ausführen: mongod –config /usr/local/etc/mongod.conf

```
Zuvor hatte ich eine ältere Version installiert. Ich werde die Verknüpfungen aufheben.
```

```shell
$ brew unlink mongodb
Unlinking /usr/local/Cellar/mongodb/3.0.7... 11 Symlinks entfernt

$ mongod --version
db version v4.4.3
Build Info: {
    "version": "4.4.3",
    "gitVersion": "913d6b62acfbb344dde1b116f4161360acd8fd13",
    "modules": [],
    "allocator": "system",
    "environment": {
        "distarch": "x86_64",
```

```
        "target_arch": "x86_64"
    }
}
```

Führen Sie dann `mongod` aus, um den MongoDB-Server zu starten. Beim ersten Start wurde jedoch gemeldet, dass `/data/db` nicht existiert. Wir erstellen ein Verzeichnis, `~/mongodb`, um die Datenbankdateien dort zu speichern.

```
$ mongod --dbpath ~/mongodb
```

Ausgabe:

```
{"t":{"$date":"2021-03-11T18:17:32.838+08:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","msg
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"STORAGE",  "id":4615611, "ctx":"initandlis
{"t":{"$date":"2021-03-11T18:17:32.842+08:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlis
{"t":{"$date":"2021-03-11T18:17:32.843+08:00"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlis
...
```

Es ist ersichtlich, dass alles im `JSON`-Format vorliegt. MongoDB speichert alle Daten in `JSON`-Format. Öffnen Sie anschließend einen weiteren Terminal-Tab.

```
$ mongo
MongoDB Shell Version v4.4.3
Verbindung zu: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implizite Sitzung: Sitzung { "id" : UUID("4f55c561-70d3-4289-938d-4b90a284891f") }
MongoDB Server Version: 4.4.3
---
Der Server hat beim Starten diese Warnungen generiert:
        2021-03-11T18:17:33.743+08:00: Zugriffskontrolle ist für die Datenbank nicht aktiviert. Lese- u
        2021-03-11T18:17:33.743+08:00: Dieser Server ist an localhost gebunden. Externe Systeme können
        2021-03-11T18:17:33.743+08:00: Soft rlimits zu niedrig
        2021-03-11T18:17:33.743+08:00:         aktueller Wert: 4864
        2021-03-11T18:17:33.743+08:00:         empfohlenes Minimum: 64000
---
---
```

```
        Aktivieren Sie den kostenlosen cloud-basierten Überwachungsdienst von MongoDB, der dann Metriken

        Die Überwachungsdaten werden auf einer MongoDB-Website mit einer eindeutigen URL verfügbar sein

Um die kostenlose Überwachung zu aktivieren, führen Sie den folgenden Befehl aus: `db.enableFreeMonitori
Um diese Erinnerung dauerhaft zu deaktivieren, führen Sie den folgenden Befehl aus: `db.disableFreeMonit
```

Anschließend können Sie versuchen, Daten einzufügen und abzufragen.

```
> db.inventory.insertOne(
...    { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
    "acknowledged" : true,
    "insertedId" : ObjectId("6049ef91b653541cf355facb")
}
>
> db.inventory.find()
{ "_id" : ObjectId("6049ef91b653541cf355facb"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ],
```

Fazit

Das war’s erstmal. Später werden wir noch andere Tools ausprobieren. Was ist der Sinn hinter all dem? Wahrscheinlich, um zunächst eine Struktur zu haben. Der Anfang ist immer schwer, und wir haben gleich alles auf einmal durchgearbeitet. Das gibt uns das Vertrauen, dass wir in der Lage sind, und jetzt geht es darum, noch mehr mit diesen Programmen herumzuspielen.

Übung

- Die Schüler erkunden ähnlich wie oben.