

**Recherche**

**Couchbase Basics**

**INSY**

**4CHITT 2015/16**

**Daniel May, Martin Weber**

|  |  |
| --- | --- |
|  | **Version 0.3** |
| **Note:** | **Begonnen am 25. April 2016** |
| **Betreuer: Michael Borko** | **Beendet am 01. Mai 2016** |

Inhaltsverzeichnis

[1 Einführung 3](#_Toc449373390)

[1.1 Ziele 3](#_Toc449373391)

[1.2 Voraussetzungen 3](#_Toc449373392)

[1.3 Aufgabenstellung 3](#_Toc449373393)

[2 Ergebnisse 4](#_Toc449373394)

[2.1 Installation und Inbetriebnahme 4](#_Toc449373395)

[2.2 CLI Befehle 9](#_Toc449373396)

[2.3 API Installation und Verwendung 9](#_Toc449373397)

[2.4 Dokumentenstruktur 9](#_Toc449373398)

[2.5 Indizierung & Map-Reduce 9](#_Toc449373399)

[2.6 Erstellung und Verwendung von eigenen Views 9](#_Toc449373400)

[3 GitHub Link 10](#_Toc449373401)

[4 Zeitmanagement 10](#_Toc449373402)

[5 Literaturverzeichnis 12](#_Toc449373403)

# Einführung

Nachdem NoSQL im Unterricht besprochen wurde, soll auch praktisch damit gearbeitet werden. Dafür wird vorerst das dokumentenorientierte, key-value basierte System Couchbase verwendet.

## Ziele

Ziel ist es sich mit der Verwendung von Couchbase vertraut zu machen, als auch ein Nachschlagewerk zur Unterstützung zu schaffen.

## Voraussetzungen

Vorausgesetzt werden die theoretischen Grundlagen zu NoSQL.

## Aufgabenstellung

„Protokollieren Sie die einzelnen Schritte zur Installation und Inbetriebnahme sowie die Verwendung einer API (Java, PHP, Python, Node.js oder C) mit Couchbase. Gehen Sie dabei näher auf das Dokumentenformat und die Abfrage (Views) der Daten ein. Verwenden Sie dabei auch die CLI um auch in der Konsole mit Couchbase arbeiten zu können.

Abzugeben ist ein detailliertes Protokoll als Teamarbeit (2er Gruppen) zur Unterstützung und Nachschlagewerk für die zukünftige Verwendung von Couchbase. Vergessen Sie nicht die wichtigen CRUD Befehle ins Protokoll aufzunehmen (SDK und CLI, Tipp: cbtransfer).

Folgende Eckpunkte sollen enthalten sein:

* Installation
* CLI Befehle
* API Installation und Verwendung
* Dokumentenstruktur (JSON, GSON)
* Erläutern der Indizierung und des Map/Reduce Vorgangs
* Erstellung und Verwendung von Views“ [BOR]

# Ergebnisse

## Installation und Inbetriebnahme

Zuerst wählt man auf der Couchbase Homepage zwischen der Enterprise Edition und der Community Edition von Couchbase Server. Da man die Enterprise Edition ohne Bezahlung verwenden kann, wird im Unterricht diese Version verwendet. Danach wählt man eine Version, üblicherweise den letzten stabilen Release, sowie das verwendete System aus und lädt die Installationsdaten herunter. Da wir Debian verwenden, erhält man ein .deb Paket. [COU.1]

Nun muss eine virtuelle Maschine erstellt werden. Für Couchbase sollte eine separate VM genutzt werden, da es relativ viele Ressourcen benötigt. Derzeit liegt die minimalen Systemanforderungen bei 4 Prozessorkernen und 4 GB RAM.

Vor der Installation sollte die aktuellste verfügbare Version von OpenSSL installiert werden. Das Debian Package heißt libssl<version>. Installiert wird das Paket, mit dem Befehl dpkg –i couchbasePackage.deb. [COU.2]

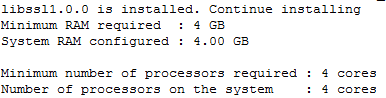


Abbildung 1: Installationsausgabe

Standardmäßig befinden sich nach der Installation alle Dateien unter /opt/couchbase.

Zusätzlich wurde ich noch darauf hingewiesen, zwei Linux Funktionen zu deaktivieren um die Performance der Datenbank zu erhöhen.

Zuerst habe ich die Transparent Huge Pages (THP) deaktiviert. Diese dienen normalerweise der Performance, indem sie zusammengehörigen pre-allokierten Speicher erstellen. Es ist jedoch empfohlen diese Funktion für Datenbanken zu deaktivieren. Nähere Informationen sind auf der Website vorhanden, die der Installer automatisch ausgibt, sollten die THP aktiviert sein. [COU.3]

Danach wurde, wie ebenfalls im Installer darauf hingewiesen, die sogenannte Swappiness auf 0 gesetzt. Dies bewirkt, dass der Arbeitsspeicher möglichst nicht auf den Festspeicher ausgelagert wird. Dies soll ebenfalls eine Performancesteigerung bei Couchbase erwirken. [COU.4]

Der Serverprozess wird automatisch nach der Installation gestartet. Wird eine Firewall verwendet, sollten die Ports, auf die im Installer hingewiesen wird, freigeschalten werden. Nun kann man unter http://<couchbaseaddress>:8091 das Setup aufrufen.

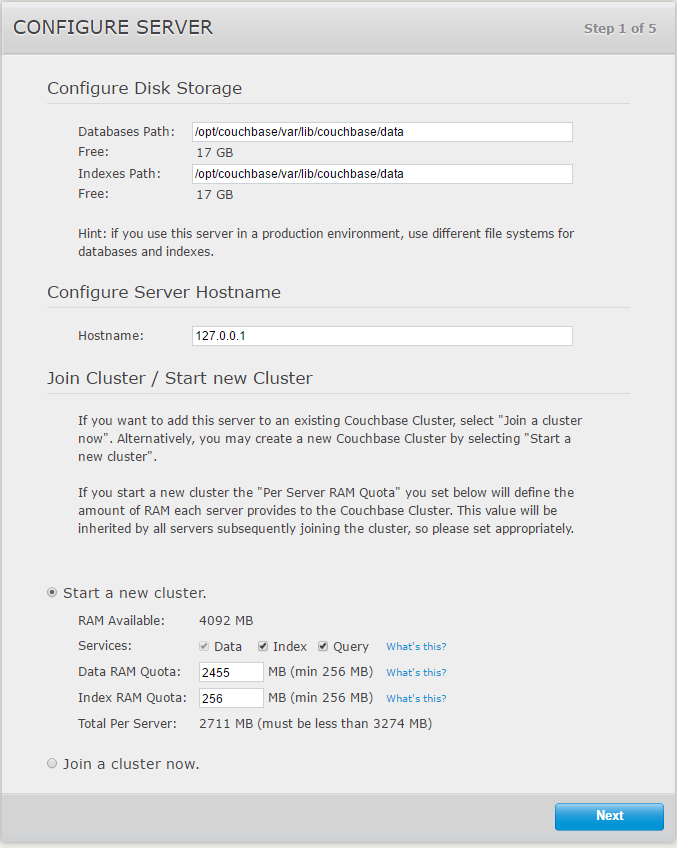


Abbildung 2: Konfiguration 1

Für nicht produzierende Umgebungen können großteils die Default-Werte akzeptiert werden, trotzdem wird das Setup hier ein wenig erläutert.

Zuerst kann bestimmt werden, wo die Datenbanken und die Indizes gespeichert werden. In einer produzierenden Umgebung sollten dies zwei unterschiedliche Speicherorte sein.

Danach kann der Hostname des Servers gesetzt werden.

Nun muss man sich entscheiden, ob diese Couchbase Installation einen neuen Cluster erstellt oder einem Cluster beitritt. Hier lässt sich gut erkennen, dass Couchbase für Hochverfügbarkeit ausgelegt ist.

Wird ein neuer Cluster erstellt, müssen zuerst die gewünschten Services konfiguriert werden. Auf jeden Fall muss diese Node (Couchbase Installation) die Daten beinhalten, zusätzlich können noch die Services Index und Query gewählt werden. Die Höhe des allokierten Hauptspeichers muss ebenfalls für die Daten, sowie den Index gesetzt werden. Diese Werte gelten für alle Nodes, die in Zukunft dem Cluster beitreten.

Tritt man einem Cluster bei, so muss man nur die IP-Adresse und die Benutzerdaten des Administrators der Node eingeben. Zusätzlich können wieder die gewünschten Services gewählt werden.

Im zweiten Schritt können Beispiele installiert werden. Diese beinhalten Datenbanken (Buckets) mit Daten und Views.

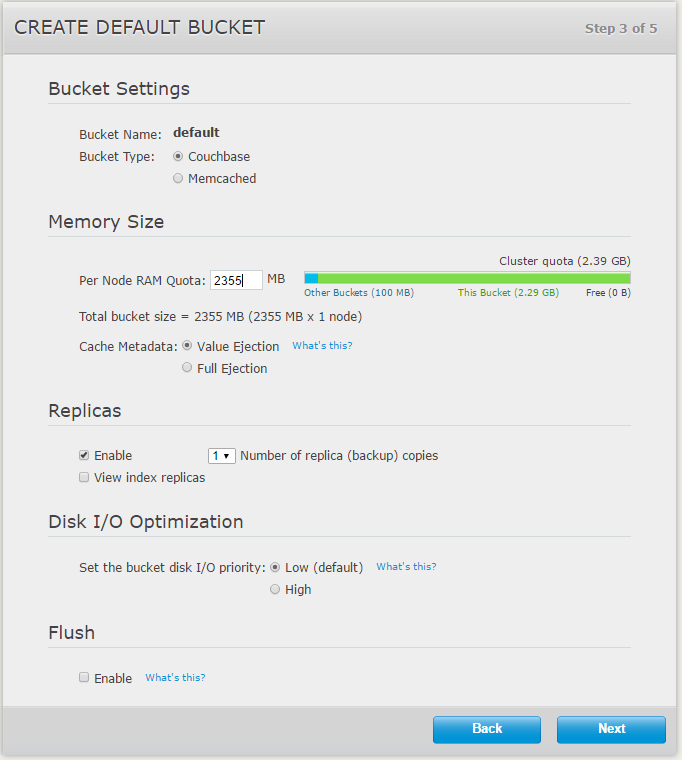


Abbildung 3: Dritter Konfigurationsschritt

Hier wird der default-Bucket konfiguriert. Dieser sollte ausschließlich für Testzwecke verwendet werden. Beim Bucket Type sollte Couchbase gewählt werden, da hier mehr Funktionen wie Persistenz, Replikation, Rebalance, XDCR und Backup verfügbar sind. Bei Memcached werden die Daten nur im RAM gespeichert, bei Couchbase zusätzlich noch persistiert. Ein Anwendungsfall für Memcached ist die Verwendung parallel zu einem RDBMS. Die häufig verwendeten Daten des RDBMS werden in einem Memcached Bucket zusätzlich zur Verfügung gestellt, um die Performance zu erhöhen.[COU.6]

Danach kann die Höhe des allokierten Hauptspeichers für diesen Bucket gesetzt werden. Dieser Wert gilt ebenfalls auf jeder Node. Nun kann das Verhalten der gecachten Metadaten gesetzt werden. Value Ejection benötigt mehr Speicherplatz, bietet aber eine bessere Performance. Full Ejection reduziert den Speicherplatz Overhead.

Danach wird die Anzahl der Backup Kopien (replicas) gesetzt. Standardmäßig werden alle Daten (Dokumente) und Funktionen repliziert. Zusätzlich könne View Index Replicas erstellt werden. Damit wird der Index auf jeder Node aus den replizierten Daten konstruiert. Dies erlaubt auch nach dem Ausfall einer Node, schnelle Queries auszuführen. [COU.7]

Für jeden Bucket kann auch die Disk I/O Priority gesetzt werden. Dementsprechend werden bei gewissen Operationen Buckets mit der Priority High bevorzugt.

Nun kann gesetzt werden ob die Daten im Bucket geflusht werden können. Das heißt ob der Bucket geleert werden kann. In einer Produktionsumgebung sollte diese Option nicht gesetzt werden.

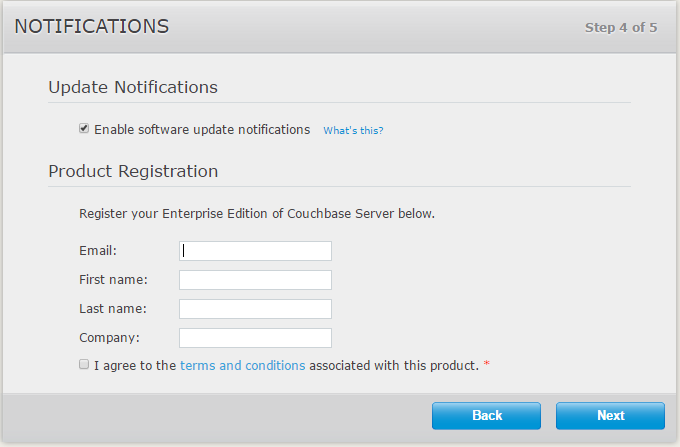


Abbildung 4: Benachrichtigungen

Nun wird das Benachrichtigungsverhalten gesetzt.

Zuletzt muss noch ein Administrator Account angelegt werden. [COU.5]

## CLI Befehle

## API Installation und Verwendung

Da wir mit Java die meiste Erfahrung sammeln konnten, wird die Installation und Verwendung der Java SDK beschrieben.

Zuerst sollte man sich unter dem Punkt „Overview“ die passende SDK Version zur Couchbase Version suchen.

Unter „Download and API reference“ findet man die gewünschte SDK Version als Download, sowie die zugehörige API. Zur Installation kann man entweder ein Maven Projekt verwenden und die Dependencies wie beschrieben eintragen oder man lädt ein Archiv herunter und bindet die JAR Files in den Build Path ein. [COU.8]

### Verbindung

Cluster cluster = CouchbaseCluster.*create*("192.168.38.140");

Bucket bucket = cluster.openBucket();

Zuerst muss man sich zu dem Cluster verbinden und danach kann man einen Bucket öffnen. Gibt man keine Argumente bei dem Cluster an, so wird eine Verbindung zu localhost aufgebaut. In einer produzierenden Umgebung sollte man mindestens zwei IP Adressen angeben, damit im Fehlerfall auf einer anderen Node weitergearbeitet werden kann. Dies geschieht automatisch mit den mitgelieferten Libraries, sodass sich der Programmierer nicht darum kümmern muss. Mehrerer IP-Adressen werden entweder als Liste oder als mehrere Parameter angegeben. [COU.8]

Da bei den IP Adressen ein Reverse DNS Lookup durchgeführt wird, sollte man einen Hosts Eintrag am ausführenden Betriebssystem anlegen. Ansonsten dauert das Lookup, vor allem auf Windows Clients, länger als 5 Sekunden, was der Standard-Timeout Wert ist. Deshalb schlägt die Verbindung fehl. Möglich wäre es auch das Timeout zu verändern, jedoch ist der Eintrag im Hosts File praktischer, da die Verbindung auch schneller aufgebaut wird. [HUB]

Danach lässt sich ein Bucket öffnen. Ohne angegebene Argumente öffnet man den Default Bucket. Man kann den zu öffnenden Bucket auch auswählen, sowie ein eventuelles Passwort dafür spezifizieren. [COU.8]

Von CouchbaseCluster, Bucket und ClusterEnvironment sollten jeweils nur eine Instanz erstellt werden.

Das ganze lässt sich natürlich auch asynchron durchführen. Grundsätzlich ist die gesamte API asynchron umgesetzt, die synchronen Methoden sind lediglich Wrapper um die asynchronen Methoden.



Abbildung 5: Asynchrone Verbindung [COU.8]

Beim Verbindungsabbau kann entweder der ganze Cluster geschlossen werden, was gleichzeitig alle geöffneten Buckets schließt, oder es werden die Buckets einzeln geschlossen, damit mit dem Cluster noch weitergearbeitet werden kann. [COU.8]

System.***out***.println("disconnect");

cluster.disconnect();

### Administratives

Mit einem ClusterManager lassen sich administrative Aufgaben auch über die Java SDK ausüben.

ClusterManager clusterManager = cluster.clusterManager("Administrator", "secretpassword");

**for** (BucketSettings bs : clusterManager.getBuckets())

System.***out***.println(bs);

Beispielsweise kann man die Einstellungen eines Buckets auslesen oder verändern. Dieser Code erzeugt zum Beispiel folgenden Output:



Abbildung 6: BucketSettings

Folgende Möglichkeiten bestehen mittels ClusterManager:

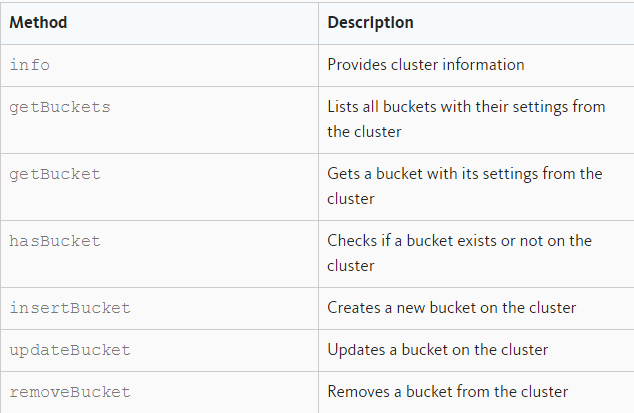


Abbildung 7: Methoden des ClusterManagers

Sollten Standardwerte bei Timeouts und ähnlichen Konfigurationen nicht ausreichen, benötigt man eine CouchbaseEnvironment.

CouchbaseEnvironment env = DefaultCouchbaseEnvironment.*builder*().connectTimeout(10000).build();

Cluster cluster = CouchbaseCluster.*create*(env, "192.168.38.140");

Soll zum Beispiel die Zeitdauer des Connection Timeouts statt 5 Sekunden 10 sein, konfiguriert man die Umgebung wie oben beschrieben und gibt die Umgebung dann an die create Methode weiter. [COU.8]

### Create

Mit der SDK können Buckets und Dokumente erzeugt werden.

Für einen Bucket muss man zuerst alle Einstellungen definieren und dann kann man ihn erstellen.

BucketSettings bucketSettings = **new** DefaultBucketSettings.Builder().type(BucketType.***COUCHBASE***)

.name("famous-persons").password("") .quota(100).replicas(1).indexReplicas(**false**) .enableFlush(**false**).build();

clusterManager.insertBucket(bucketSettings);

Anschließend verbindet man sich zu diesem Bucket und kann auch schon Dokumente hochladen.

JsonObject user = JsonObject.*empty*().put("firstname", "Walter").put("lastname", "White")

.put("job", "chemistry teacher").put("age", 50);

JsonDocument doc = JsonDocument.*create*("walter", user);

JsonDocument response = bucket.upsert(doc);

Zuerst erstellt man das gewünschte JSON Objekt oder Array. Danach erstellt man daraus ein JSON Dokument mit einer ID und dem Inhalt. Mit insert oder upsert lassen sich die Daten hochladen. Insert liefert jedoch einen Fehler, wenn das Dokument bereits vorhanden ist. [COU.8]

### Read

Wie bereits vorhin erwähnt lassen sich Informationen wie die verfügbaren Buckets usw. über einen ClusterManager auslesen.

Die erste Variante des Lesens ist, ein ganzes Dokument herunterzuladen.

JsonDocument loadedFromId = bucket.get("batman");

System.***out***.println(loadedFromId.content());

Dabei spezifiziert man die ID des Dokuments und danach erhält man ein JSON Dokument.

Die nächste Variante ist, eine View zu verwenden.

ViewResult result = bucket.query(ViewQuery.*from*("dev\_firstnames", "firstnames"));

**for** (ViewRow row : result)

System.***out***.println(row);

In einer Java Client Applikation wird dies die häufigste Variante sein, da sie die schnellen, im Vorhinein erstellten, Views verwendet.

Danach gibt es nach die Möglichkeit einer N1QL Query. Diese ist wie SQL aufgebaut, jedoch ist es eine langsame Variante Daten abzufragen. Bevor eine solche Query überhaupt abgesetzt werden kann muss ein Index erstellt werden.

bucket.query(N1qlQuery.*simple*("CREATE PRIMARY INDEX ON `famous-persons` USING GSI"));

Danach kann eine Query abgesetzt werden. Diese kann entweder als reiner Text oder mittels eigenen Methoden erstellt werden. Mit Text:

N1qlQueryResult queryResult = bucket.query(N1qlQuery.*simple*("SELECT job FROM `famous-persons`"));

**for** (N1qlQueryRow row : queryResult)

System.***out***.println(row);

Oder mit Hilfsmethoden:

N1qlQueryResult queryResult2 = bucket.query(*select*("age").from("`famous-persons`"));

**for** (N1qlQueryRow row : queryResult2)

System.***out***.println(row);

Dazu wird der statische Import (com.couchbase.client.java.query.Select.select) benötigt. [COU.8]

### Update

Wie bereits vorhin erwähnt lassen sich Einstellungen von Buckets über einen ClusterManager verändern.

Bereits erwähnt wurde ebenfalls die upsert Methode, die ein Dokument auf dem Server überschreibt oder auch neu hinzufügt. Zusätzlich gibt es noch die Replace Methode, die einen Fehler wirft, sollte das Dokument noch nicht vorhanden sein. [COU.8]

### Delete

Möglich ist das Löschen einzelner Dokumente:

JsonDocument doc = bucket.remove("batman");

Dazu gibt man lediglich eine ID an. Außerdem lassen sich mit dem ClusterManager ganze Buckets löschen.

clusterManager.removeBucket("famous-persons");

Dazu muss man auch nur den Namen des Buckets angeben. [COU.8]

## Dokumentenstruktur

## Indizierung & Map-Reduce

## Erstellung und Verwendung von eigenen Views

# GitHub Link

<https://github.com/dmay-tgm/Couchbase-Basics>

# Zeitmanagement

Daniel May übernimmt die Installation, API Installation inkl. Verwendung und die Dokumentenstruktur. Martin Weber übernimmt die CLI Befehle, Indizierung und Map/Reduce, sowie die Erstellung und Verwendung von Views. Korrekturen werden jeweils auch am anderen Teil durchgeführt.

Der Aufwand wird wie folgt geschätzt:

|  |  |  |
| --- | --- | --- |
| **Arbeitsteil** | **Durchführender** | **geschätzter Aufwand** |
| Installation | Daniel May | 0.5 h |
| CLI Befehle | Martin Weber | 1.5 h |
| API Installation und Verwendung | Daniel May | 1.5 h |
| Dokumentenstruktur (JSON, GSON) | Daniel May | 0.5 h |
| Indizierung & Map/Reduce | Martin Weber | 0.5 h |
| Erstellung und Verwendung von Views | Martin Weber | 1 h |
| Korrekturarbeiten | D. May & M. Weber | 0.5 h/Person |
| **Gesamt** | **D. May & M. Weber** | **6.5 h** |

Der tatsächliche Aufwand ist wie folgt aufgeschlüsselt:

**Daniel May:**

|  |  |  |
| --- | --- | --- |
| **Durchgeführte Arbeit** | **Datum** | **tatsächlicher Aufwand** |
| Installation | 25.04.2016 | 1.25 h |
| API Installation und Verwendung | 01.05.2016 | h |
| Dokumentenstruktur (JSON, GSON) | 01.05.2016 | h |
| Korrekturen | 01.05.2016 | 0.1 h |
| **Gesamt** | **25.04.2016** | **1.25 h** |

**Martin Weber:**

|  |  |  |
| --- | --- | --- |
| **Durchgeführte Arbeit** | **Datum** | **tatsächlicher Aufwand** |
| CLI Befehle | 25.04.2016 | h |
| Indizierung & Map/Reduce | 25.04.2016 | h |
| Erstellung und Verwendung von Views | 25.04.2016 | h |
| Korrekturen | 25.04.2016 | h |
| **Gesamt** | **25.04.2016** | **h** |

**Gesamt:**

|  |  |
| --- | --- |
| **Durchführender** | **tatsächlicher Aufwand** |
| Daniel May | 1.25 h |
| Martin Weber | h |
| **Gesamt** | **h** |

# Literaturverzeichnis

[BOR] Michael Borko.   
Couchbase [Online]. Available at: <https://elearning.tgm.ac.at/mod/assign/view.php?id=40870>   
[abgerufen am 25.04.2016]

[COU.1] Couchbase.   
Downloads [Online]. Available at: <http://www.couchbase.com/nosql-databases/downloads>   
[abgerufen am 25.04.2016]

[COU.2] Couchbase (Version 4.1).   
Quick installation and setup [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/getting-started/installing.html>   
[abgerufen am 25.04.2016]

[COU.3] Couchbase (Version 4.1).   
Transparent Huge Pages (THP) [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/install/thp-disable.html>   
[abgerufen am 25.04.2016]

[COU.4] Couchbase (Version 4.1).   
Swap space and kernel swappiness [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/install/install-swap-space.html>   
[abgerufen am 25.04.2016]

[COU.5] Couchbase (Version 4.1).   
Initial server setup using UI [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/install/init-setup.html#topic12527>   
[abgerufen am 25.04.2016]

[COU.6] Couchbase (Version 4.1).   
Buckets [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/architecture/core-data-access-buckets.html>   
[abgerufen am 25.04.2016]

[COU.7] Couchbase (Version 4.1).   
View replication [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/indexes/mapreduce-view-replication.html>   
[abgerufen am 25.04.2016]

[COU.8] Couchbase (Version 4.1).   
Java SDK 2.2 (inkl. Unterpunkte) [Online]. Available at: <http://developer.couchbase.com/documentation/server/4.1/sdks/java-2.2/java-intro.html>   
[abgerufen am 25.04.2016]

[HUB] hubo3085632 (September 2015).   
JavaClient 2.x connection`s warning [Online]. Available at: <https://forums.couchbase.com/t/javaclient-2-x-connection-s-warning/5227>   
[abgerufen am 25.04.2016]