

Getting to know



Cassandra

by Michelle Darling
mdarlingcmt@gmail.com

August 2013

Agenda:

- What is Cassandra?
- Installation, CQL3
- Data Modelling
- Summary

Only 15 min to cover these, so please hold questions til the end, or [email me](#) :-) and I'll summarize Q&A for everyone.

Unfortunately, no time for:

- DB Admin
 - Detailed Architecture
 - Partitioning / Consistent Hashing
 - Consistency Tuning
 - Data Distribution & Replication
 - System Tables
- App Development
 - Using Python, Ruby etc to access Cassandra
 - Using Hadoop to stream data into Cassandra

What is Cassandra?



“Fortuneteller of Doom”

from Greek Mythology. Tried to warn others about future disasters, but no one listened. Unfortunately, she was 100% accurate.

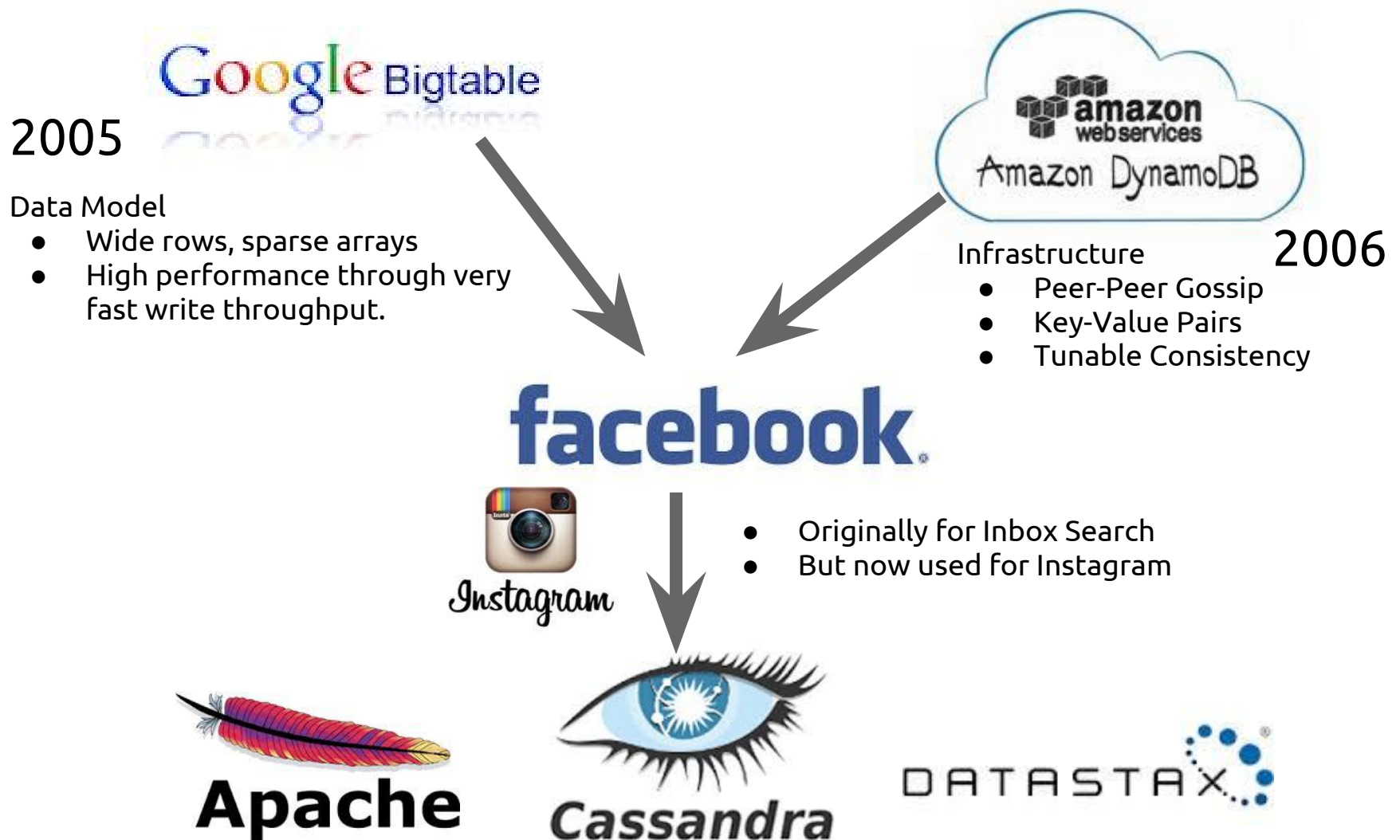
NoSQL Distributed DB

- Consistency - A__ID
- Availability - High
- Point of Failure - none
- **Good for Event Tracking & Analysis**
 - Time series data
 - Sensor device data
 - Social media analytics
 - Risk Analysis
 - Failure Prediction

Rackspace: “Which servers are under heavy load and are about to crash?”



The Evolution of Cassandra



2008: Open-Source Release / 2013: Enterprise & Community Editions

Other NoSQL vs. Cassandra

NoSQL Taxonomy:

- Key-Value Pairs
 - Dynamo, Riak, Redis
- Column-Based
 - BigTable, HBase, **Cassandra**
- Document-Based
 - MongoDB, Couchbase
- Graph
 - Neo4J

Big Data Capable

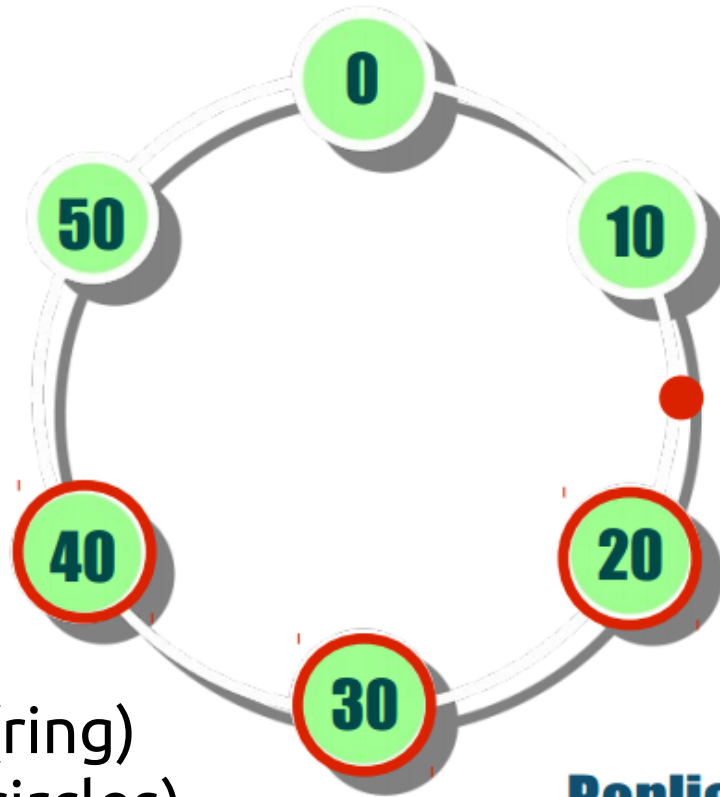
C* Differentiators:

- Production-proven at Netflix, eBay, Twitter, 20 of Fortune 100
- “Clear Winner” in Scalability, Performance, Availability

Cassandra – affectionately “C*” – is an open source, distributed store for structured data that scales-out on cheap, commodity hardware and stays up **even when things get really bad.**

-- *DataStax*

Architecture



Partitioner:

Consistent Hashing

Key: “www.google.com”

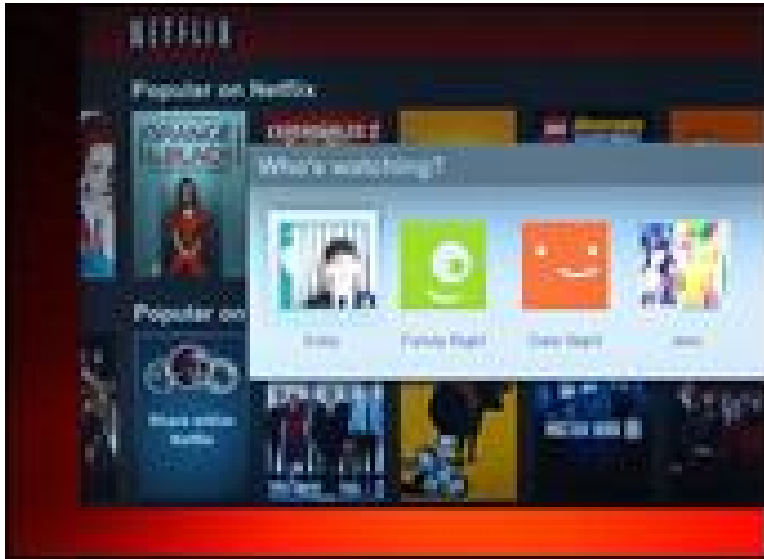
md5(“www.google.com”)

14

Replication Factor = 3

- Cluster (ring)
- Nodes (circles)
- Peer-to-Peer Model
- Gossip Protocol

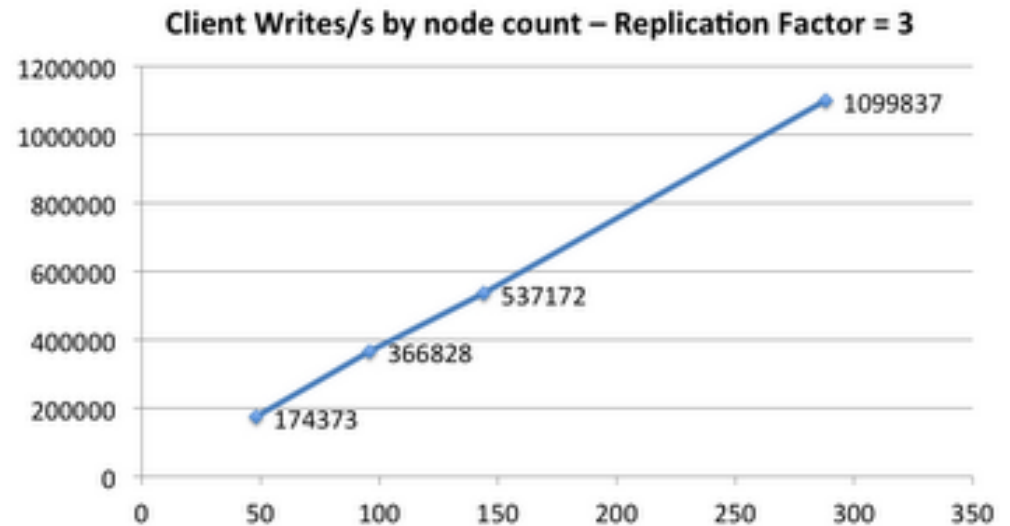
“Netflix foretells 'House of Cards' success with Cassandra big data engine Big data algorithms helped Netflix map the viewing patterns of its customers”



Netflix Streaming Video

- [Personalized Recommendations](#) per family member
- Built on Amazon Web Services (AWS) + Cassandra

Scale-Up Linearity



Cloud installation using **EC2**

- **Amazon Web Services (AWS)**
- **Elastic Compute Cloud (EC2)**
 - Free for the 1st year! Then pay only for what you use.
 - Sign up for AWS EC2 account: Big Data University [Video 4:34 minutes](#),
- **Amazon Machine Image (AMI)**
 - Preconfigured installation template
 - Choose: **“DataStax AMI for Cassandra Community Edition”**
 - Follow these **very good** step-by-step [instructions](#) from DataStax.
 - AMIs also available for CouchBase, MongoDB (make sure you pick the free tier community versions to avoid monthly charge\$\$!!!).

AWS EC2 Dashboard



Services ▾ Edit ▾

Michelle Darling ▾ Oregon ▾ Help ▾

EC2 Dashboard

Events
Tags

INSTANCES

Instances
Spot Requests
Reserved Instances

IMAGES

AMIs
Bundle Tasks

ELASTIC BLOCK STORE

Volumes
Snapshots

NETWORK & SECURITY

Security Groups
Elastic IPs
Placement Groups
Load Balancers

Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

0 Running Instances	0 Elastic IPs
0 Volumes	0 Snapshots
2 Key Pairs	0 Load Balancers
0 Placement Groups	2 Security Groups

Optimize your resources' cost, performance and security with **AWS Trusted Advisor**

Hide

Create Instance

To start using Amazon EC2 you will need to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon):
This service is operating normally

Scheduled Events

US West (Oregon):

No events

Account Attributes

Supported Platforms

EC2-VPC

Default VPC

vpc-ae2e05c6

Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

Popular AMIs on AWS Marketplace

Debian GNU/Linux

Provided by Debian
Rating ★★★★★
Free Software, pay only for AWS usage
View all Operating Systems

Couchbase Server - Community Edition

DataStax AMI Setup

Request Instances Wizard



CHOOSE AN AMI

INSTANCE DETAILS

CREATE KEY PAIR

CONFIGURE FIREWALL

REVIEW

The bookmark that was activated refers to the AMI below. Please review...

AMI Details

Image Id: ami-544e6e11
Owner: 620704289608
Manifest: datastax-us-west-1/datastax_clustering_ami_2.4.manifest.xml
Platform:  Other Linux
Architecture: x86_64
Root Device Type: instance-store

Attached Block Devices

Device Name	Volume Size
sdb	ephemeral0
sdc	ephemeral1

DataStax AMI Setup

The screenshot shows the AWS Management Console with the 'Request Instances Wizard' open. The wizard is at the 'CHOOSE AN AMI' step. The 'Number of Instances' is set to 1, and the 'Availability Zone' is 'No Preference'. The 'Advanced Instance Options' section is expanded, showing 'Kernel ID' as 'Use Default', 'RAM Disk ID' as 'Use Default', and 'Monitoring' as 'Enable CloudWatch detailed monitoring for this instance (additional charges will apply)'. The 'User Data' field is set to 'as text' and contains the following content: `--clustername Michelle --totalnodes 1 --version community`. A tooltip is visible over the 'User Data' field, showing the same content. The 'Termination Protection' is set to 'None', and the 'IAM Role' is 'None'. The 'Continue' button is highlighted.

Request Instances Wizard

CHOOSE AN AMI | INSTANCE DETAILS | CREATE KEY PAIR | CONFIGURE FIREWALL | REVIEW

Number of Instances: 1 | Availability Zone: No Preference

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choose to enable CloudWatch Detailed Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: Use Default | RAM Disk ID: Use Default

Monitoring: ☐ Enable CloudWatch detailed monitoring for this instance (additional charges will apply)

User Data: `--clustername Michelle --totalnodes 1 --version community`

☒ as text
☐ as file

(Use shift+enter to insert a new line)

base64 encoded ☐ Prevention against accidental termination ☐

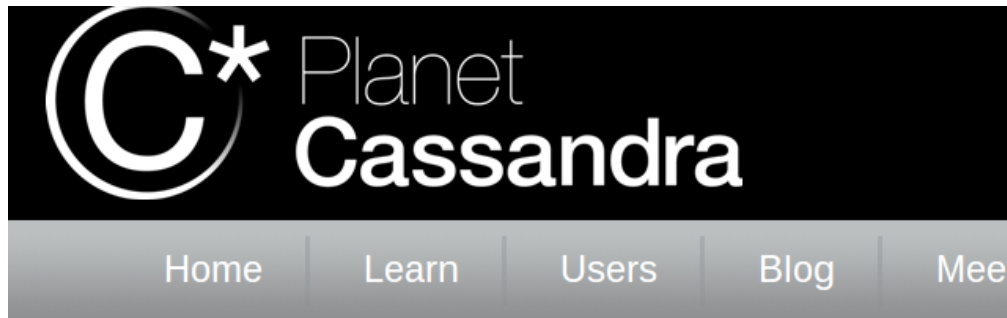
Termination Protection: ☐

IAM Role: None

< Back | Continue >

“Roll your Own” Installation

DataStax Community Edition



DataStax Community Edition

DSC

Operating System

DSC Version



Getting Started

DataStax Community Edition is a free packaged distribution of Apache Cassandra™ made available by DataStax. There's no faster, easier way to get started with Cassandra than to download, install, and use DataStax Community Edition.

- **Install instructions**
For Linux, Windows, MacOS:

<http://www.datastax.com/2012/01/getting-started-with-cassandra>

- **Video:** “Set up a 4-node Cassandra cluster in under 2 minutes”

<http://www.screenr.com/5G6>

Invoke CQLSH, CREATE KEYSPACE

```
./bin/cqlsh
```

```
cqlsh> CREATE KEYSPACE big_data
```

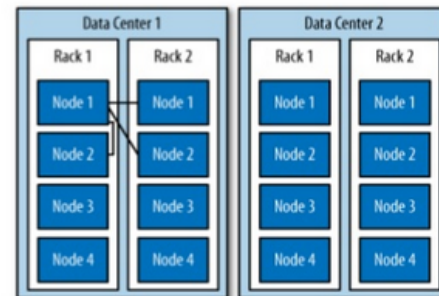
```
... with strategy_class = 'org.apache.cassandra.  
locator.SimpleStrategy'
```

```
... with strategy_options:replication_factor='1';
```

```
cqlsh> use big_data;
```

```
cqlsh:big_data>
```

- SimpleStrategy - returns the nodes that are next to each other on the ring.





Tip: Skip Thrift -- use CQL3

Thrift RPC

```
// Your Column
Column col = new Column(ByteBuffer.wrap("name".
getBytes()));
col.setValue(ByteBuffer.wrap("value".getBytes()));
col.setTimestamp(System.currentTimeMillis());
// Don't ask
ColumnOrSuperColumn cosc = new ColumnOrSuperColumn();
cosc.setColumn(col);
// Prepare to be amazed
Mutation mutation = new Mutation();
mutation.setColumnOrSuperColumn(cosc);
List<Mutation> mutations = new ArrayList<Mutation>();
mutations.add(mutation);
Map mutations_map = new HashMap<ByteBuffer, Map<String,
List<Mutation>>>();
Map cf_map = new HashMap<String, List<Mutation>>>();
cf_map.set("Standard1", mutations);
mutations_map.put(ByteBuffer.wrap("key".getBytes()),
cf_map);
cassandra.batch_mutate(mutations_map,
consistency_level);
```

CQL3

- Uses cqlsh
- "SQL-like" language
- Runs on top of Thrift RPC
- Much more user-friendly.

Thrift code on left
equals this in CQL3:

```
INSERT INTO (id, name)
VALUES ('key',
'value');
```

CREATE TABLE

#thisisahashtag



```
cqlsh:big_data> create table user_tags (  
... user_id varchar,  
... tag varchar,  
... value counter,  
... primary key (user_id, tag)  
...):
```



- **TABLE user_tags:** "How many times has a user mentioned a hashtag?"
- **COUNTER** datatype - Computes & stores counter value at the time data is written. This optimizes query performance.

UPDATE TABLE

SELECT FROM TABLE

```
cqlsh:big_data> UPDATE user_tags SET  
value=value+1 WHERE user_id = 'paul' AND tag =  
'cassandra'
```

```
cqlsh:big_data> SELECT * FROM user_tags
```

user_id	tag	value
paul	cassandra	1

DATA MODELING

A Major Paradigm Shift!



RDBMS	Cassandra
Structured Data, Fixed Schema	Unstructured Data, Flexible Schema
"Array of Arrays" 2D: ROW x COLUMN	"Nested Key-Value Pairs" 3D: ROW Key x COLUMN key x COLUMN values
DATABASE	KEYSPACE
TABLE	TABLE a.k.a COLUMN FAMILY
ROW	ROW a.k.a PARTITION. Unit of replication.
COLUMN	COLUMN [Name, Value, Timestamp]. a.k.a CLUSTER. Unit of storage. Up to 2 billion columns per row.
FOREIGN KEYS, JOINS, ACID Consistency	Referential Integrity not enforced, so A_CID. BUT relationships represented using COLLECTIONS.

Rows

Columns

Column names

	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7902	17-DEC-88	800.00	300.00	20
7499	ALLEN	SALESMAN	7698	20-FEB-88	1600.00	300.00	30
7521	WARD	SALESMAN	7698	22-FEB-88	1250.00	500.00	30
7566	JONES	MANAGER	7839	02-APR-88	2975.00		20

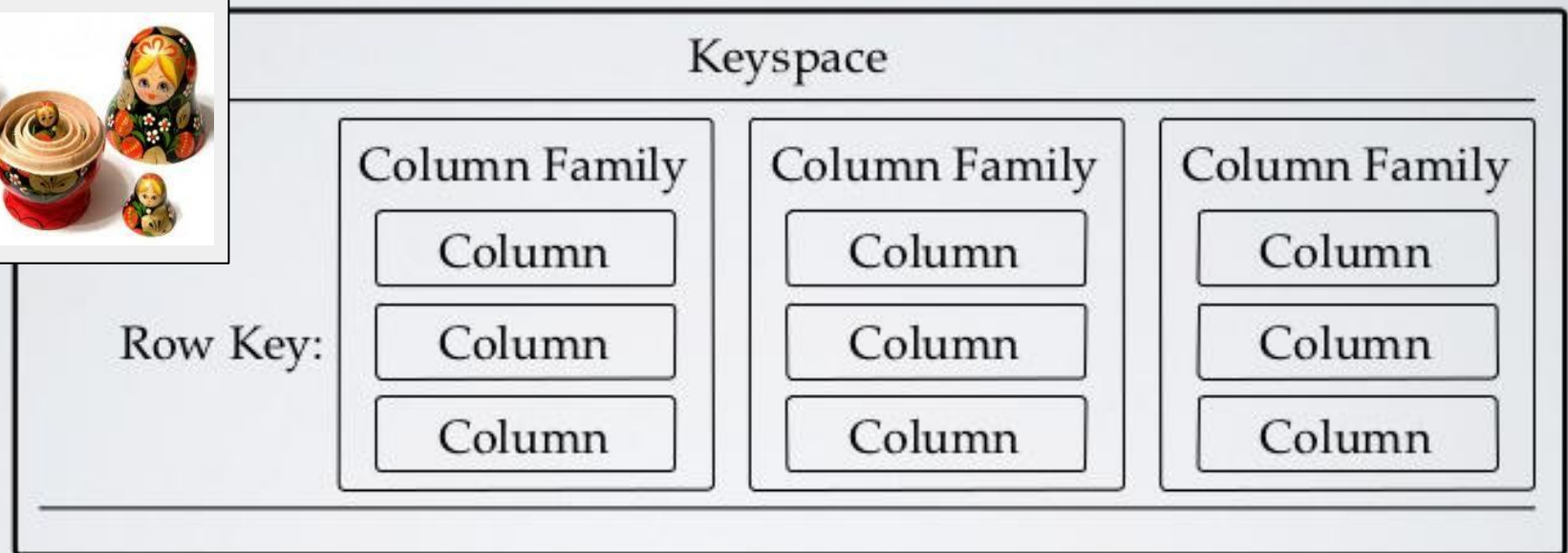
Column not allowing nulls

Column allowing nulls



RDBMS
2D: Rows
x columns

Cassandra
3D+: Nested Objects



(Column Family and Table mean the same.)

Example:

“Twissandra” Web App

Twitter-Inspired
sample application
written in Python +
Cassandra.

- Play with the app:
twissandra.com
- Examine & learn
from the [code](#) on
GitHub.

Features/Queries:

- **Sign In, Sign Up**
- **Post Tweet**
- **Userline** (User's tweets)
- **Timeline** (All tweets)
- **Following** (Users being followed by user)
- **Followers** (Users following this user)

Twissandra.com vs Twitter.com

Twissandra

[Home](#)

[Public](#)

[Find Friends](#)

[Sign out of tom](#)

Post Tweet

[tom](#) Try Twissandra/you will learn a lot!

[tom](#) Shout out to UCSC Big Data class!

[tom](#) Hello

[jim](#) Sorry test not interested.

[tom](#) Hello Every!

[tom](#) Hello, How are you today?

[tom](#) hello from aj

[tom](#) hi

[tom](#) Hi Boston.

[tom](#) Hello Tim

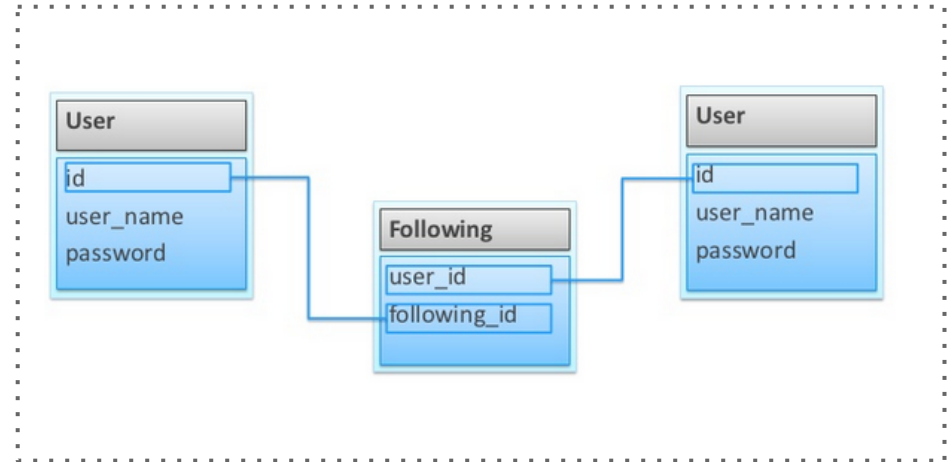
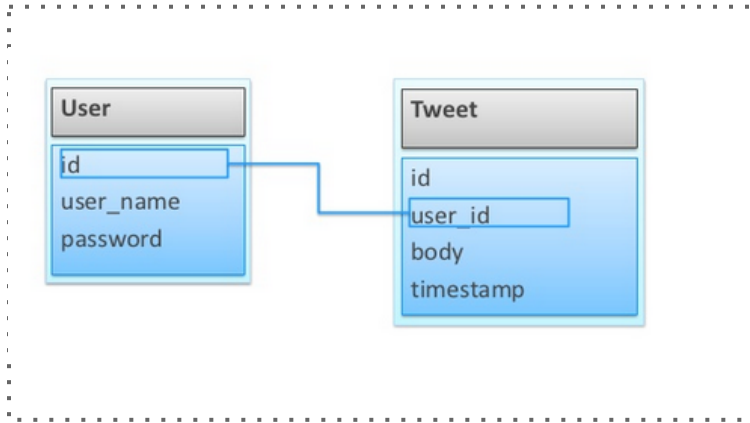
[tom](#) hi



Twissandra is an example project, created to learn and demonstrate how to use Cassandra. Running the project will present a website that has similar functionality to Twitter.

View the code for this site [on GitHub](#)

Twissandra - RDBMS Version

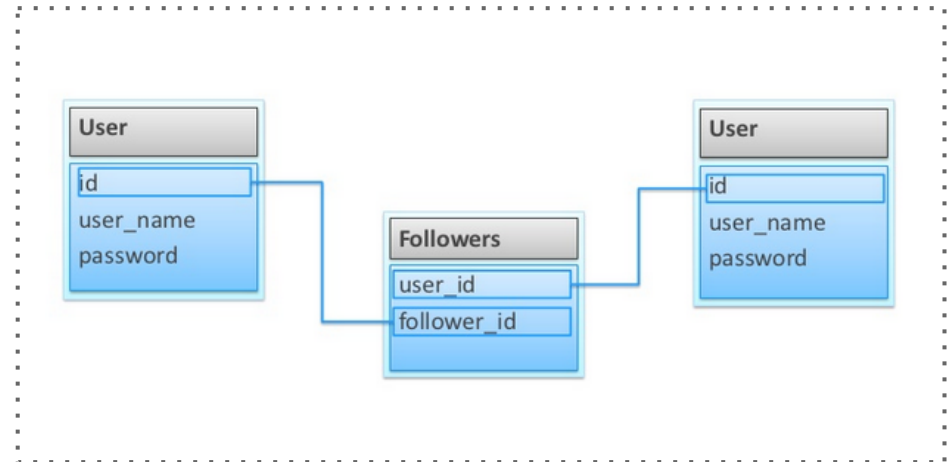


Entities

- USER, TWEET
- FOLLOWER, FOLLOWING
- FRIENDS

Relationships:

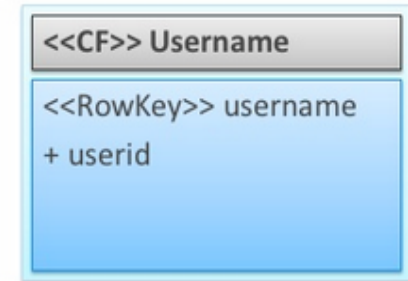
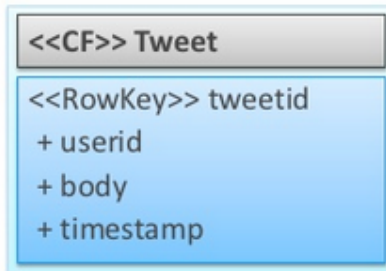
- USER has many TWEETS.
- USER is a FOLLOWER of many USERS.
- Many USERS are FOLLOWING USER.





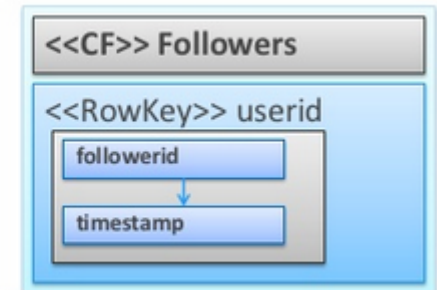
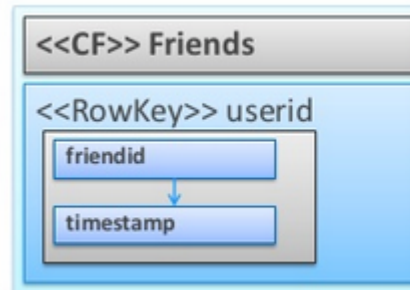
Twissandra - Cassandra Version

Tip: Model tables to mirror queries.



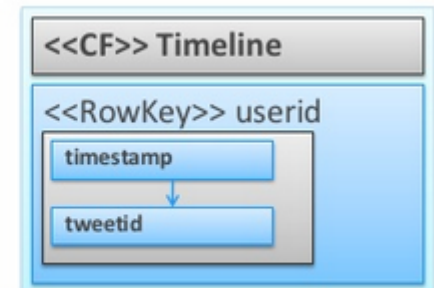
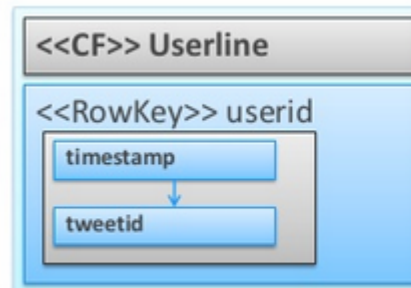
TABLES or CFs

- TWEET
- USER, USERNAME
- FOLLOWERS, FOLLOWING
- USERLINE, TIMELINE



Notes:

- Extra tables mirror queries.
- Denormalized tables are “pre-formed” for faster performance.

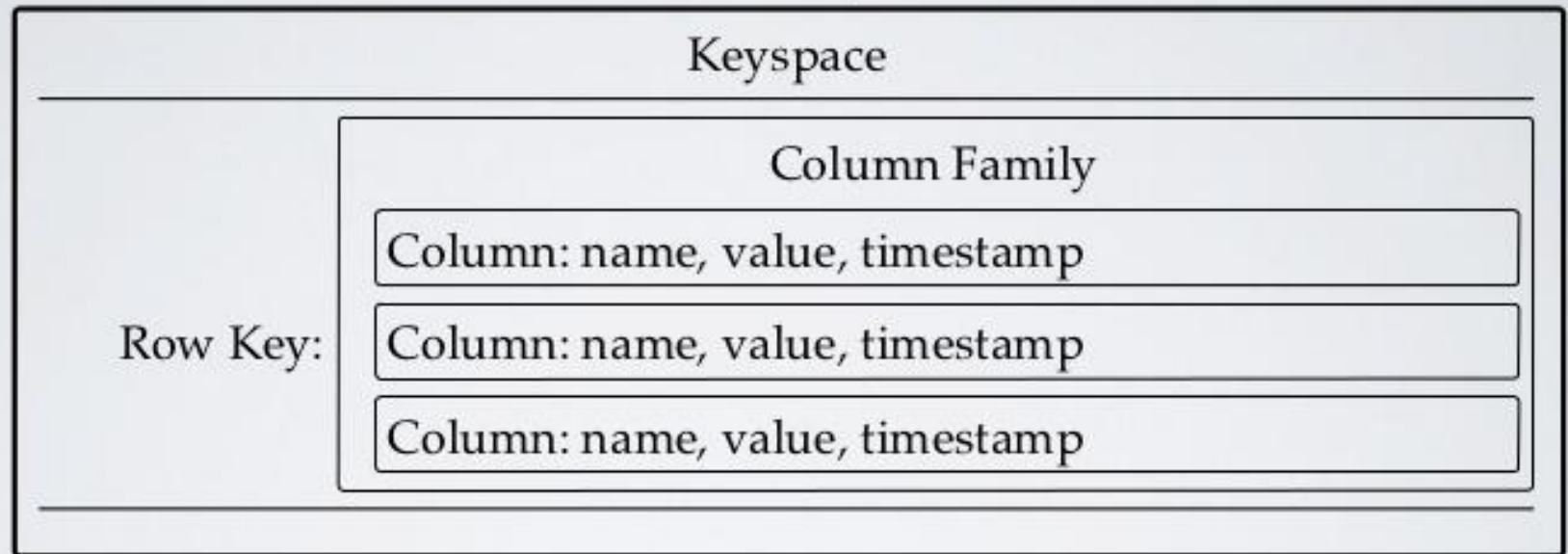




Tip: Remember, Skip Thrift -- use CQL3

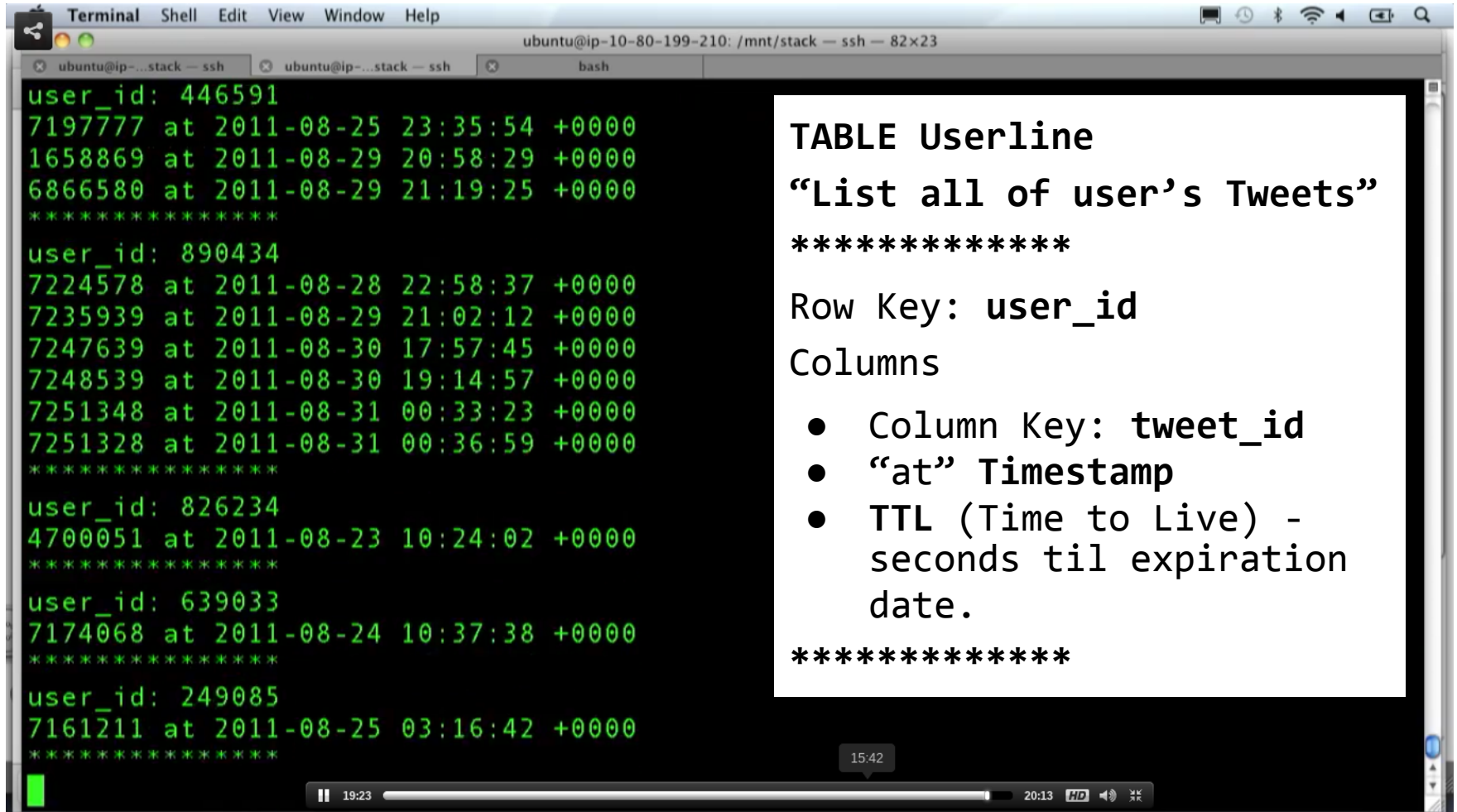
Inside the ~~Column Family~~.

TABLE



(Also TTL Columns)

What does C* data look like?



The image shows a terminal window with a dark background and green text. It displays data for a table named 'Userline'. The data is organized into sections, each starting with a 'user_id' followed by a list of rows. Each row contains a tweet_id, a timestamp, and a TTL value. The rows are separated by asterisks. A white box on the right side of the terminal provides a description of the table and its columns.

```
user_id: 446591
7197777 at 2011-08-25 23:35:54 +0000
1658869 at 2011-08-29 20:58:29 +0000
6866580 at 2011-08-29 21:19:25 +0000
*****

user_id: 890434
7224578 at 2011-08-28 22:58:37 +0000
7235939 at 2011-08-29 21:02:12 +0000
7247639 at 2011-08-30 17:57:45 +0000
7248539 at 2011-08-30 19:14:57 +0000
7251348 at 2011-08-31 00:33:23 +0000
7251328 at 2011-08-31 00:36:59 +0000
*****

user_id: 826234
4700051 at 2011-08-23 10:24:02 +0000
*****

user_id: 639033
7174068 at 2011-08-24 10:37:38 +0000
*****

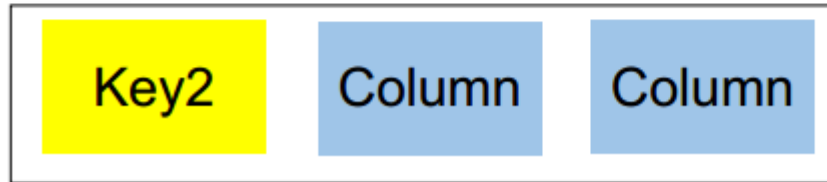
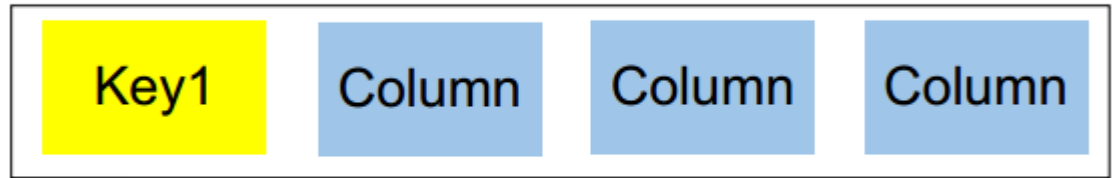
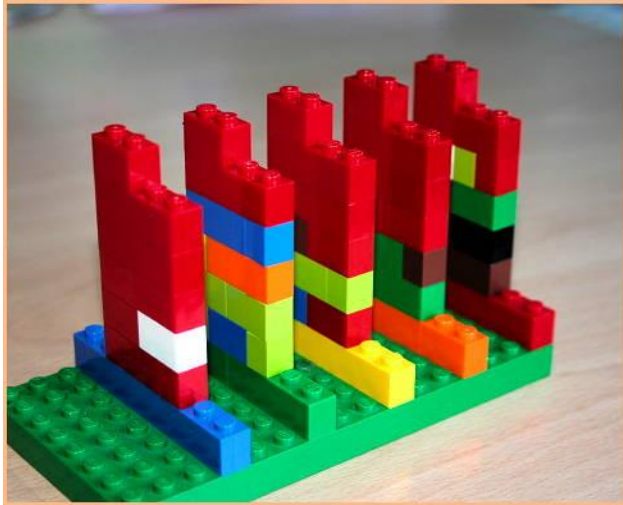
user_id: 249085
7161211 at 2011-08-25 03:16:42 +0000
*****
```

TABLE Userline
“List all of user’s Tweets”

Row Key: **user_id**
Columns

- Column Key: **tweet_id**
- “at” Timestamp
- TTL (Time to Live) - seconds til expiration date.

Cassandra Data Model = LEGOs?



*Flexible
Schema*

Stored sorted by column key/name →

Row key1	Column Key1	Column Key2	Column Key3	...
	Column Value1	Column Value2	Column Value3	
⋮				

Stored sorted by row key ↓



Summary:

- **Go straight from SQL to CQL3;** skip Thrift, Column Families, SuperColumns, etc
- **Denormalize tables** to mirror important queries. Roughly 1 table per imp't query.
- **Choose wisely:**
 - Partition Keys
 - Cluster Keys
 - Indexes
 - TTL
 - Counters
 - Collections

See DataStax [Music Service](#) Example

- **Consider hybrid approach:**

- 20% - RDBMS for highly structured, OLTP, ACID requirements.
- 80% - Scale Cassandra to handle the rest of data.

Remember:

- **Cheap:** storage, servers, OpenSource software.
- **Precious:** User AND Developer Happiness.

Resources

C* Summit 2013:

- [Slides](#)
- Cassandra at eBay Scale ([slides](#))
- Data Modelers Still Have Jobs - Adjusting For the NoSQL Environment ([Slides](#))
- Real-time Analytics using Cassandra, Spark and Shark [slides](#)

- Cassandra By Example: Data Modelling with CQL3 [Slides](#)
- DATASTAX C*OLLEGE CREDIT: DATA MODELLING FOR APACHE CASSANDRA [slides](#)

I wish I found these 1st:

- How do I Cassandra? [slides](#)
- Mobile version of DataStax web docs ([link](#))