

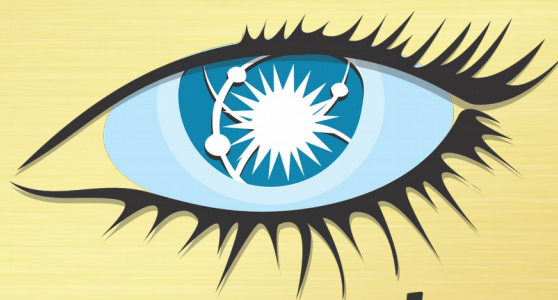
**Linux Málaga**

@linux\_malaga www.linux-malaga.org



**Centrologic**

**Taller de**



***cassandra***

Bienvenido - Welcome - Witam

**Juan Miguel Taboada Godoy**



**@juanmitaboda**

<https://www.linkedin.com/in/juanmitaboda>

**Juan José Soler Ruiz**



**@soleronline**

<http://es.linkedin.com/in/soleronline>

## Juan Miguel Taboada Godoy ( 1980 - ... )

1996 – Primer ordenador y primera LAN (cable coaxial)

1999 – Universidad de Málaga y **Linux Málaga**

2001 – Grupo de investigación **GEB.uma.es** (4 años)  
Cluster computación +20 nodos (**OpenMosix**)

2002 – Presidente de Asociación **Málaga Wireless**

2003 – Beca en **Neurociencia** en SUNY  
Teleruta (Ministerio de Fomento – 2 años)

2005 – Autónomo:

- Nace **Centrologic**
- Polonia (2 años)
- Likindoy (Axaragua + **Junta Andalucía**)

2008 – Responsable Sistemas en PontGrup

2011 – Adquisición Datos en Bética Fotovoltáicas

2012 – **SAFECLON** y SCRUM/KANBAN

2013 – **Executive MBA**

2014 – **Aeronáutica: Django + AngularJS**

2015 – **Industria: Likindoy + Big Data**

## Juan José Soler Ruiz ( 1982 - ... )

2001 – CFGS Administración Sistemas Informáticos

2003 – Primer premio en el concurso “Javier Benjumea”

2003 – Montaje y configuración  
de “Cluster Heterogéneo De Computadoras”  
bajo SO Red Hat 7.2.

2005 – STEA Telemática

2007 – Primer CRM en PHP

2010 – Administrador de BBDD / Programador  
Web en Bética Fotovoltáicas

2010 – Opositometro

2012 – **Centrologic** (externo)

2013 – Dailymarkets

2013 – CRM en Python/Django

2014 – **Centrologic**







# ¿Por qué?





# Likindoy

The monitoring company

Software de adquisición masiva de datos:

- 1 dato (Fecha+Valor) cada minuto por sensor
- 1000 sensores por dispositivo
- 20 dispositivos por nodo
- 40 nodos por cliente

**800.000 registros por minuto**  
(8.000.000 tomas por minuto)

- 48M por hora
- 1.152M por día

**0,4 Billones por año por cliente**





- La gestión de datos crece de momento EXPONENCIALMENTE
- SEAGATE anuncia que el año 2016 será el año del Zettabyte
- 1 ZB equivale al espacio que ocupa 2 billones de años de música
- 1 ZB = 1024 exabytes = casi 1.1 trillones de Gbytes
- 1 zB = 1 sextillón de bytes
- Ahora estamos en la época del **quintillón** de bytes...

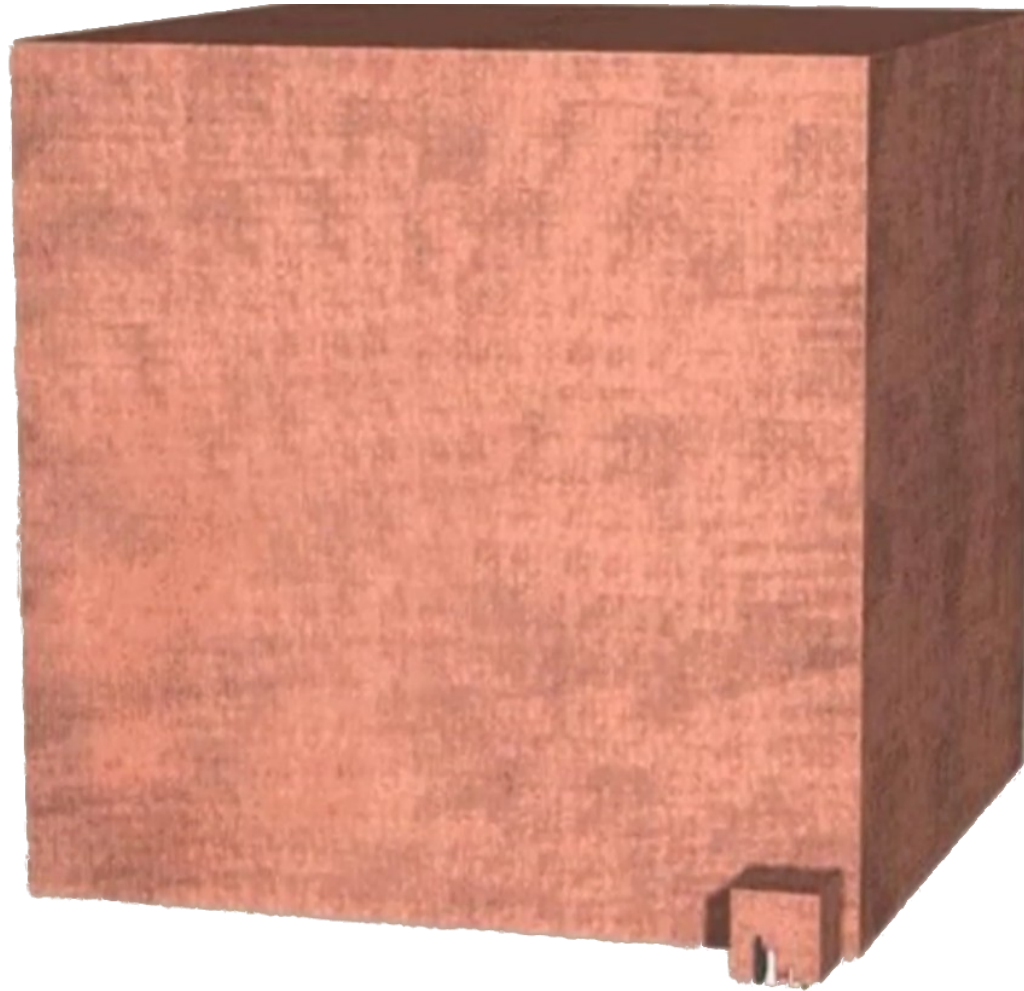


# ¿Cuanto es un quintillón?

- Una única **gota de agua** contiene:  
**1.7 quintillones** de moléculas de agua.
- La distancia de la **Via Láctea hasta Andrómeda** es de:  
2 millones de años luz  
**18,87 quintillones de kilómetros**  
11,73 quintillon miles
- **La tierra** completa contiene unos:  
**1.234 quintillones de litros de agua**  
326 quintillon gallons of water
- Si cortamos **la tierra por la mitad**, la sección tendría un área aproximada de:  
**1.275 quintillones de centímetros cuadrados**
- ¿Cuanto es un quintillón de céntimos o peniques de dólar?



# ¿Cuanto es un quintillón... de peniques?



1.000.067.088.384.000.000 peniques

1 quintillón, 67 trillones, 88 billones, 384 millones de peniques

Un cubo de **8,32 kilómetros** de lado





Eric Brewer (2000)

C: Consistency → Consistencia

A: Availability → Disponibilidad

P: Partition tolerance → Tolerancia al particionado

**Sólo puedes llegar a 2 de las 3**

Es imposible para un sistema de cómputo distribuido garantizar simultáneamente la **consistencia**, la **disponibilidad** y ser tolerante al **particionado** de los datos (separación y distribución).





# Teorema de CAP

Lo que la mayoría  
**piensa** que tiene

Siempre se puede  
leer y escribir

A

C

P

Todos los clientes  
ven siempre lo mismo

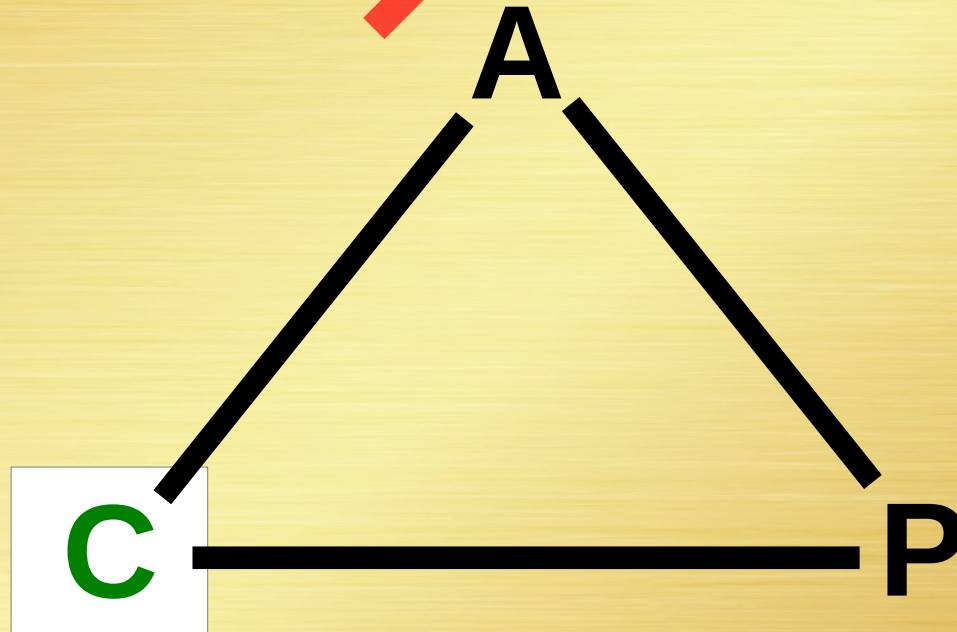
Funcionará incluso  
cuando haya errores  
en el sistema



# Teorema de CAP

Lo que la mayoría  
de **verdad** tiene

~~Siempre se puede  
leer y escribir~~



Todos los clientes  
ven siempre lo mismo

~~Funcionará incluso  
cuando haya errores  
en el sistema~~





Lo cierto es que todos buscamos la **disponibilidad (A)**

**Pero ... ¡¡¡ tenemos que elegir entre... !!!**

Escalabilidad (P)

y

Consistencia (C)



## ACID

**A:** Atomicidad

**C:** Consistencia

**I:** Aislamiento (Isolation)

**D:** Durabilidad

En grandes sistemas ocurre que:  
~~Disponibilidad~~ y ~~Rendimiento~~





## BASE

**BA:** Básicamente disponible

**S:** Flexible (Soft state)

**E:** Consistencia eventual

Da **menos** importancia a la **consistencia**  
en **pro** de la tolerancia al **particionado**  
aparece la **consistencia eventual**



## ¿Qué es la consistencia eventual?

Que ... eventualmente será consistente

Podemos introducir un dato y que no esté disponible inmediatamente después

Convergencia natural a la consistencia





# Teorema de CAP

Siempre se puede  
leer y escribir

**A**

**CA**

MySQL    Postgres  
Oracle    SQLServer

**AP**

Dynamo    **Cassandra**  
SimpleDB    CouchDB

Todos los clientes  
ven siempre lo mismo

**C**

**P**

Funcionará incluso  
cuando haya errores  
en el sistema

**CP**

MongoDB    Redis  
MemcacheDB    Hbase  
Berkely DB



# Teorema de CAP

## CA

MySQL  
Postgres  
Oracle  
SQLServer

Aster Data  
Greenplum  
**Vertica**

## AP

Dynamo  
Voldemort  
Tokyo Cabinet  
KAI

Cassandra  
SimpleDB  
CouchDB  
Riak



## CP

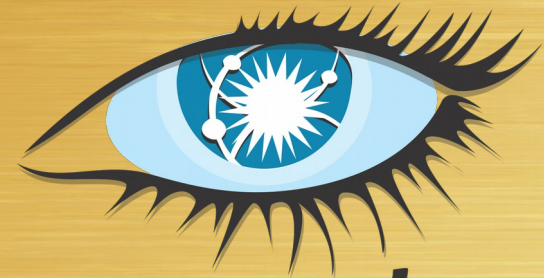
**BigTable**  
Hypertable  
**HBase**

MongoDB  
Terrastore  
**Scalaris**

Berkely DB  
MemcacheDB  
Redis







# cassandra

Disponibilidad continua

Simplicidad en la gestión entre servidores  
Sin un único punto de fallo

Escalabilidad lineal

- Si 2 nodos procesan 100 transacciones/seg
- 4 nodos procesan 200 transacciones/seg
- 8 nodos procesan 400 transacciones/seg

Sistema **descentralizado** (sin master)

Relaciones por grupo: Nodo → Datacenter → Cluster  
Replicación personalizada

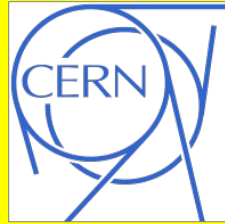




# ¿Quién usa Cassandra?

Fuente: <http://planetcassandra.org/companies/>

CALL OF DUTY



NBCOMCAST

Alcatel-Lucent

agoda

CREDIT SUISSE

The New York Times

accenture  
High performance. Delivered.

KASPERSKY Lab

OPERA software

Aol.

NBCUniversal



reddit



Adobe



Spotify

vmware



THOMSON REUTERS



UBISOFT

Disney



FedEx

SanDisk



rackspace



PayPal



Centrologic



Expedia



Sony Entertainment

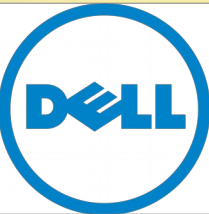
softonic



Symantec

CISCO

ebay



ERICSSON

GitHub

NETFLIX



BlackBerry

IBM

despegar.com

ING



LEONEL MESSI

SHAZAM



Telefonica



Instagram

Fast beautiful photo sharing



MuleSoft™  
connecting the new enterprise



Microsoft



unity



Centrologic

18/28



Linux  
M6laga

Big Data





# ¿Quién usa Cassandra?



Al finalizar la última iteración del LHC en el que se descubrió el “**Bosón de Higgs**”, el CERN almacenaba más de **100Pbytes**

El LHC del CERN arrancó en **Abril de 2015** en busca de la **Supersimetría** (capaz de producir 1Pbytes/segundo)

**1Petabyte = 1.000 Terabytes = 1 Millón de Gigabytes**

-----

**NETFLIX**

Ejecuta 235 clusters separados, con un total de 7.000 nodos.  
1 Millón de escrituras por segundo (factor 3)

-----



Ecosistema para desarrolladores de juegos que teniendo problemas con MongoDB migraron a Cassandra.



**Centrologic**

19/28



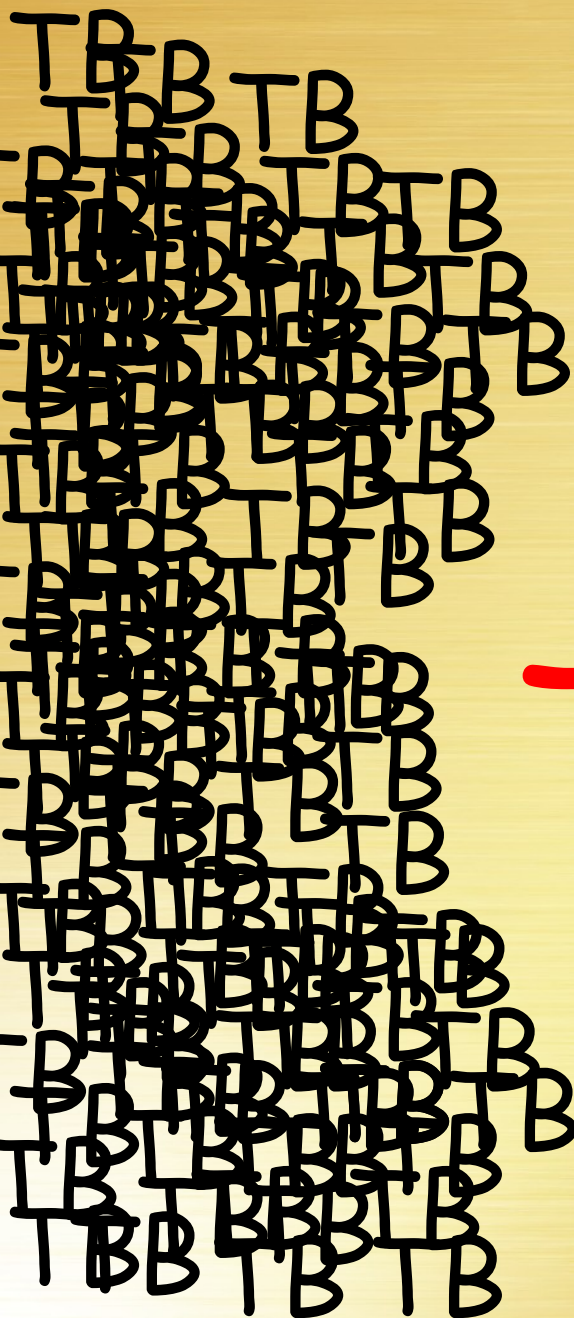
**Linux  
Mology**

**Big Data**





...BREAK...



¡Hey tío!, somos 1023,  
¿te vienes y nos hacemos un Peta?

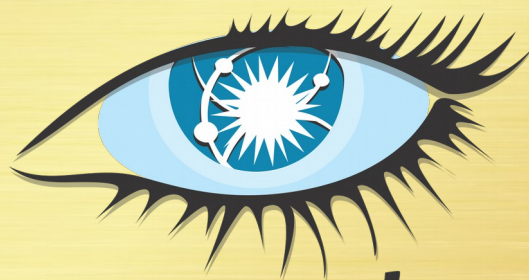
TB

TB

¡Hecho!



# Guía de CQL



*cassandra*



Centrologic

21/28



Linux  
M6

Big Data





¿Cómo conectarnos desde la línea de comando?

```
$ cqlsh
```

Creando un Keyspace

```
> CREATE KEYSPACE centrologic  
    WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : 2 };
```

Ver Keyspaces

```
> DESCRIBE KEYSPACES;
```

Usar un Keyspace

```
> USE centrologic;
```

Ver las tablas de un Keyspace

```
> DESCRIBE TABLES;
```



## Creación de tablas

```
> CREATE TABLE test1 (
    id uuid,
    ciudad text,
    pais text,
    poblacion int
    latitud float,
    longitud float,
    PRIMARY KEY (id)
);
```

## Insertar datos

```
> INSERT INTO centrologic.test1 (id, ciudad, pais, poblacion, latitud, longitud)
    VALUES (uuid(), 'Malaga', 'Spain', 572947, 36.716667, -4.416667);

> INSERT INTO centrologic.test1 (id, ciudad, pais, poblacion, latitud, longitud)
    VALUES (uuid(), 'Madrid', 'Spain', 3207247, 40.418889, -3.691944);
```





## Seleccionar datos

```
> SELECT * FROM centrologic.test1;
```

## Filtrar datos

```
> SELECT * FROM centrologic.test1 WHERE ciudad = 'Malaga';
```

Para poder hacer búsqueda por un campo este debe tener un índice

## Crear índices secundarios

```
> CREATE INDEX ON centrologic.test1 (ciudad);
```

## Volver a consultar datos

```
> SELECT * FROM centrologic.test1 WHERE ciudad = 'Malaga';
```



## Modificar el tipo de un campo

> **ALTER TABLE** centrologic.test1 **ALTER** ciudad **TYPE** varchar;

## Agregar un campo a un tabla

> **ALTER TABLE** centrologic.test1 **ADD** distrito varchar;

> **ALTER TABLE** centrologic.test1 **ADD** codigopostal **LIST** <text>;

## Eliminar un campo de una tabla

> **ALTER TABLE** centrologic.test1 **DROP** distrito;

## Vaciar un tabla

> **TRUNCATE TABLE** centrologic.test1;





## Borrar un índice

Vemos la información de la tabla para obtener el nombre del índice

```
> DESCRIBE TABLE centrologic.test1;
```

Eliminamos el índice

```
> DROP INDEX centrologic.test1_cuidad_idx;
```

## Borrar una tabla

```
> DROP TABLE centrologic.test1;
```

## Borrar un keyspace

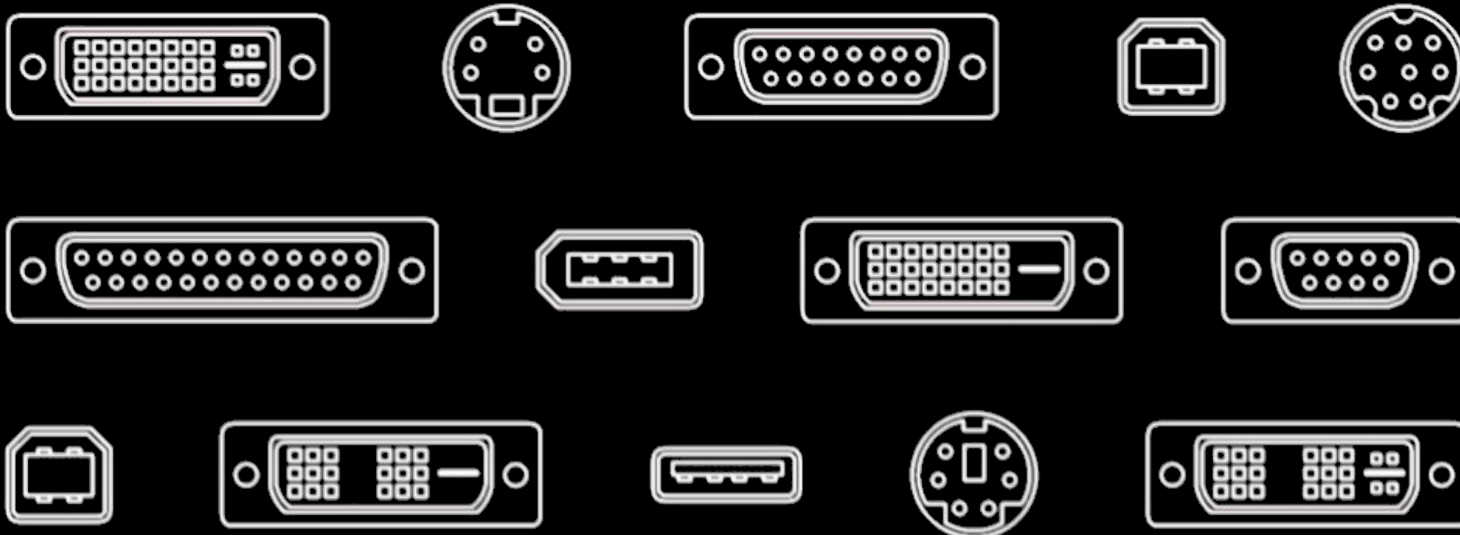
```
> DROP TABLE centrologic;
```

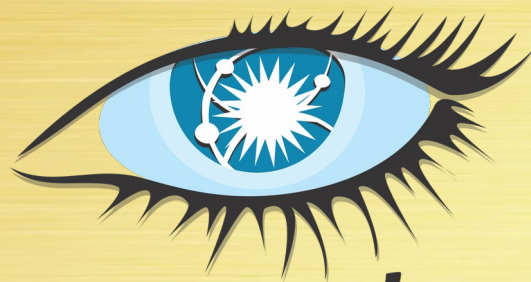
## Modificar factor de replicación según el datacenter

```
> ALTER KEYSPACE centrologic  
  WITH REPLICATION = { 'class': 'SimpleStrategy', 'datacenter1': 3, 'datacenter2': 2 };
```



# WHEN IN DOUBT TRY ANOTHER HOLE





*cassandra*

**Muchas  
Gracias**

Thank you - Dziękuję



**Juan Miguel Taboada Godoy**  
<http://www.centrologic.com>

[@juanmitaboada](#)

<https://www.linkedin.com/in/juanmitaboada>



**Juan José Soler Ruiz**

[@soleronline](#)

<http://es.linkedin.com/in/soleronline>



**Likindoy**  
The monitoring company



**Centrologic**

**Linux Málaga**

[@linux\\_malaga](#) [www.linux-malaga.org](http://www.linux-malaga.org)