



Features of key data services in IBM Bluemix PaaS



After you complete this unit, you should understand:

- The features of the following data services in IBM Bluemix PaaS:
 - Cloudant NoSQL DB
 - dashDB
 - SQL Database
 - Time Series Database

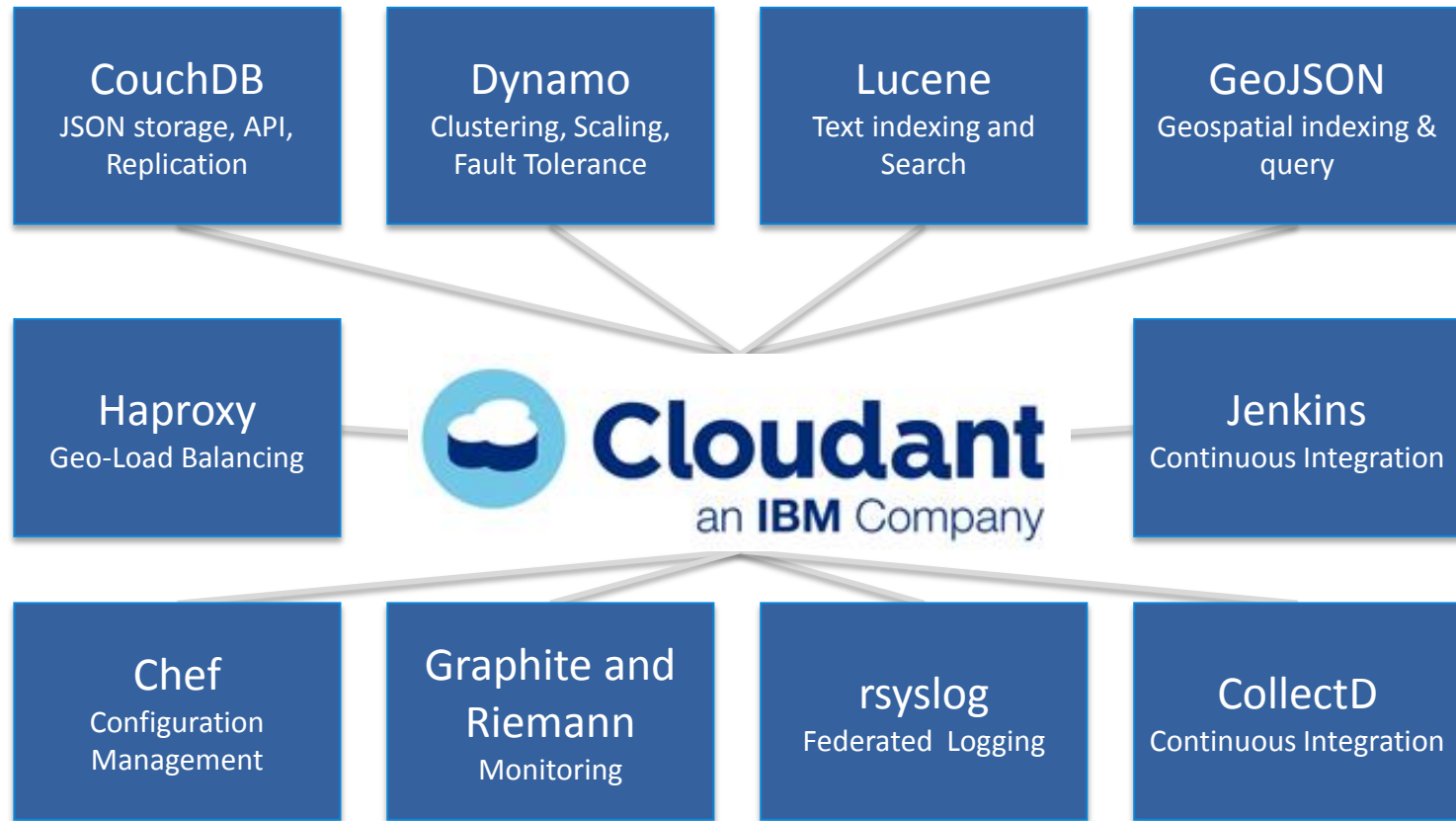
Cloudbant NoSQL service

- A fully managed NoSQL database as a service
- Transactional JSON “document” database with RESTful API
- Can spread data across data centers and devices for scale and high availability (HA)
- Ideal for apps that require:
 - Massive, elastic scalability
 - High availability
 - Geo-location services
 - Full-text search
 - Occasionally connected users



Build More. Grow More. Sleep More.

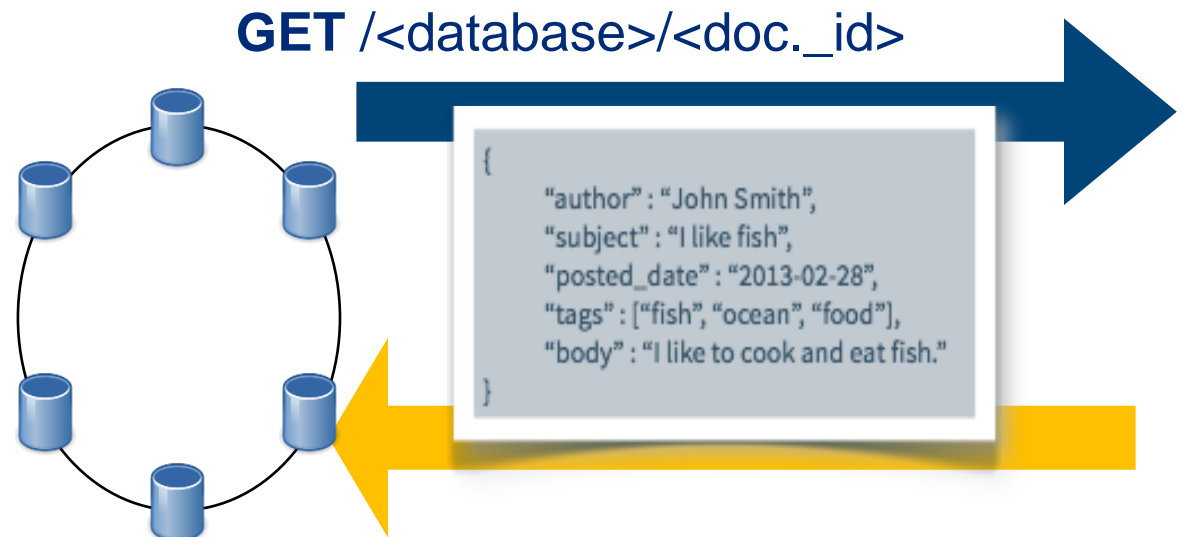
Cloudbant DNA



Cloudbant combines the best Open Source technology and thinking to create the most scalable, flexible, always-on DBaaS for big mobile and the Internet of Things.

Cloudbant HTTP RESTful API

- CRUD operations, document retrieval through indexes, search
- Agnostic to programming language
- Apache CouchDB compatibility



Highlights of the Cloudant API



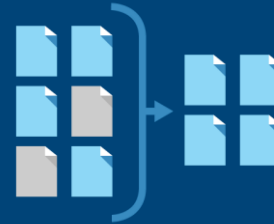
JSON documents

- Create
- Read
- Update
- Delete



Primary index

- Exists for every database out of the box
- Find docs by their primary key → `_id`



Secondary indexes

- Built by using MapReduce
- Use when you need to analyze data
- Example: count data fields, aggregate or sum results, and so on



Search

- Built by using Lucene
- Ad-hoc queries
- Find documents based on their content

Cloudbant documents: `_id` and `_rev`

- Each document has an `_id` (ID) field that is unique per database.
 - Any string can be supplied as an `_id`, but it is recommended that you allow Cloudbant to generate a UUID (universally unique identifier) for you.
- There is also a unique `_rev` (revision number) field per document.
 - This is generated by an md5 hash of the transport representation of the document.
 - *N*-prefix reflects the number of times this document has been updated.
 - Updates to existing documents must provide the latest `_rev` value; otherwise, the update request is rejected.

```
{ "_id": "7f123e23a328bd50ee123cd35452ae47",  
  "_rev": "2-3123414209",  
  "title": "IBM Cloudbant Redbook",  
  "author": "Christopher Bienko" }
```

Cloudant API: Returning a single document

HTTP GET with database name and `_id` of document. For example, to get a document with `_id` **100** from the **authors** database:

GET `https://[username].cloudant.com/authors/100`




```
{  
  "_id": "100",  
  "name": "John Smith",  
  "agent": "Mary Reid",  
  "telephone": "512-555-1212"  
}
```

Cloudant API: Inserting a document (HTTP PUT or POST)

Via **POST**: document `_id` in document body

POST `https://[username].cloudant.com/authors`




```
{
  "_id": "101",
  "name": "Mary Smith",
  "agent": "John Reid",
  "telephone": "512-555-1212"
}
```

```
{
  "ok": "true",
  "id": "101",
  "rev": "1-0af5e..."
}
```

Via **PUT**: document `_id` in URL

PUT `https://[username].cloudant.com/authors/101`



```
{
  "name": "Mary Smith",
  "agent": "John Reid",
  "telephone": "512-555-1212"
}
```

```
{
  "ok": "true",
  "id": "101",
  "rev": "1-0af5e..."
}
```

Cloudant API: Updating a document (HTTP PUT or POST)

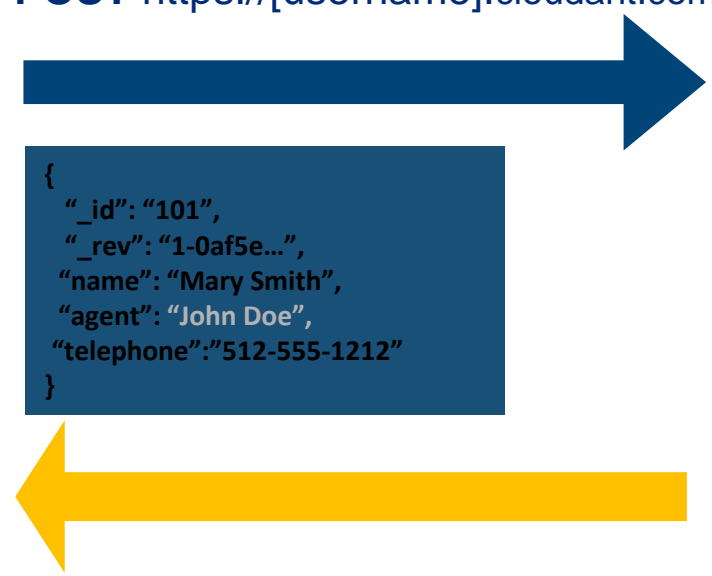
Same HTTP operations as insert: if `_id` exists, then it's an update; latest `_rev` is required or the operation will fail.

Via **POST**: document `_id` in document body

POST [https://\[username\].cloudant.com/authors](https://[username].cloudant.com/authors)


Via **PUT**: document `_id` in URL

PUT [https://\[username\].cloudant.com/authors/101](https://[username].cloudant.com/authors/101)



```
{
  "_id": "101",
  "_rev": "1-0af5e...",
  "name": "Mary Smith",
  "agent": "John Doe",
  "telephone": "512-555-1212"
}
```

```
{
  "ok": "true",
  "id": "101",
  "rev": "2-03f5e..."
}
```



```
{
  "_rev": "1-0af5e...",
  "name": "Mary Smith",
  "agent": "John Doe",
  "telephone": "512-555-1212"
}
```

```
{
  "ok": "true",
  "id": "101",
  "rev": "2-03f5e..."
}
```

Cloudant API: Deleting a document (HTTP DELETE or PUT)

Latest `_rev` is required or the operation will fail.

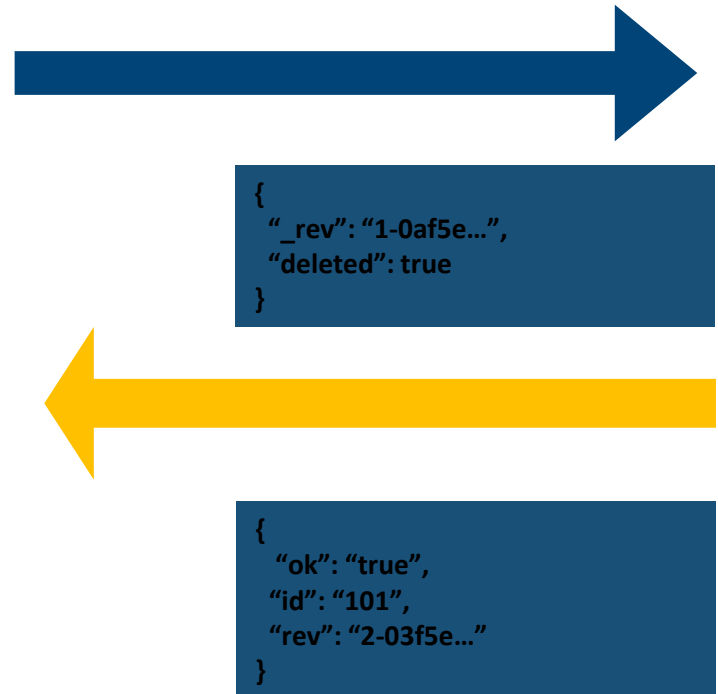
Via **DELETE**: document `_id` and `_rev` in URL

DELETE `https://[username].cloudant.com/authors/101?rev=...`



Via **PUT**: document `_id` in URL , `_rev` and `"deleted": true` in document body

PUT `https://[username].cloudant.com/authors/101`



Cloudant: Secondary indexes

- Cloudant allows the creation of secondary indexes (or views) that use MapReduce to return specific subsets of data.
 - The Map function returns a list of key-value pairs.
 - This function must be defined for a view.
 - The Reduce function reduces the list to a single value per key.
 - This function is optional.
- Map and Reduce functions for a view are:
 - Written in JavaScript
 - Stored in special documents called *design documents*

MapReduce function example

- A list of cars

<code>_id: 1 make: Audi model: A3 year: 2000 bookval: 5400</code>	<code>_id: 2 make: Audi model: A4 year: 2009 bookval: 16000</code>	<code>_id: 3 make: VW model: Golf year: 2009 bookval: 15000</code>	<code>_id: 4 make: VW model: Golf year: 2008 bookval: 9000</code>	<code>_id: 5 make: VW model: Polo year: 2010 bookval: 12000</code>
---	--	--	---	--

- A list of car makes and their book values (map function)



- Aggregated book value by make (reduce function)

```
function(key, values) {
  return sum (values);
}
```

Map results

Key	Value
Audi	5400
Audi	16000
VW	9000
VW	12000
VW	15000

Reduce results

Key	Total Book Value
Audi	21400
VW	36000

Cloudbant sync

- Native replication feature that allows you to push database access to mobile devices, remote facilities, sensors, and Internet-enabled devices
- Enables mobile and distributed apps to scale by replicating and syncing data between multiple readable and writable copies of a database even on mobile iOS and Android devices
- Simplifies large-scale mobile development by enabling you to create a single local database for every user
 - Reduces round-trip database requests with the server because when there is no network connection, the app runs off the database on the device
 - When the network is restored, local data is synced with server

dashDB



dashDB

Service	Use case	Free tier (30-day) sizing	Entitled (paid) sizing
dashDB	Data warehousing and accelerated analytics	Includes perpetual free tier up to 1 GB stored data	<ul style="list-style-type: none"> Up to 20 GB (entry) Dedicated instances from 64 GB up to 256 GB of data

- Stores relational data for querying and advanced analytics
 - Data mining
 - Predictive analytics
 - Geospatial analytics
- Powered by IBM BLU Acceleration and Netezza in-database analytics
 - IBM BLU Acceleration is fast and simple. It uses dynamic in-memory columnar technology and innovations such as actionable compression to rapidly scan and return relevant data.
 - In-database analytic algorithms integrated from Netezza bring simplicity and performance to advanced analytics.



SQL Database

SQL database

Service	Use case	Free tier (30-day) sizing	Entitled (paid) sizing
SQL Database	Web, transactional	Includes perpetual free tier up to 100 MB of stored data	<ul style="list-style-type: none"> Small: 10 GB per instance Premium: 500 GB per instance

- On-demand relational database for applications running on IBM Bluemix
 - Powered by DB2
- Provides a managed database service to handle web and transactional workloads
- Offers high availability, automated backups, and data privacy

Time Series
Database

Time Series database

Service	Use case	Free tier (30-day) sizing	Entitled (paid) sizing
Time Series Database	Internet of Things	Includes perpetual free tier up to 1 GB stored data	10 GB stored data per instance

- Highly **efficient storage**, lowering costs for storing massive amounts of time series data in the cloud
- **Time Series functions** (extended SQL) make writing applications much simpler than standard SQL, faster iteration
 - **Spatial data** also handled with highly optimized, built-in functions
- Time series offered with a fully functional, enterprise-class relational database framework (Informix)
 - **Support for both SQL and NoSQL** (JSON) within the same database
 - Allows for data movement between the two interfaces

Summary: data services in IBM Bluemix PaaS

- Cloudbant NoSQL Database service
 - Document-oriented schema-less data store optimized for horizontal scaling
- dashDB service
 - Data warehousing service for relational data, including special types such as geospatial data
 - Analyze data with SQL or advanced built-in analytics-like predictive analytics and data mining, analytics with R, and geospatial analytics
- SQL Database service
 - DB2 backed database service with high availability, automated backups and data privacy
- Time Series Database service
 - Managed data store for time-stamped Internet of Things device data
 - Optimized storage for large volumes and time-based SQL extensions