Developing Modern Applications with Db2

simplify coding

```
{
  "store"   : "json",
  "call"    : "RESTful",
  "code"    : "OData",
  "exploit" : "relational",
  "get"     : "results"
}
```

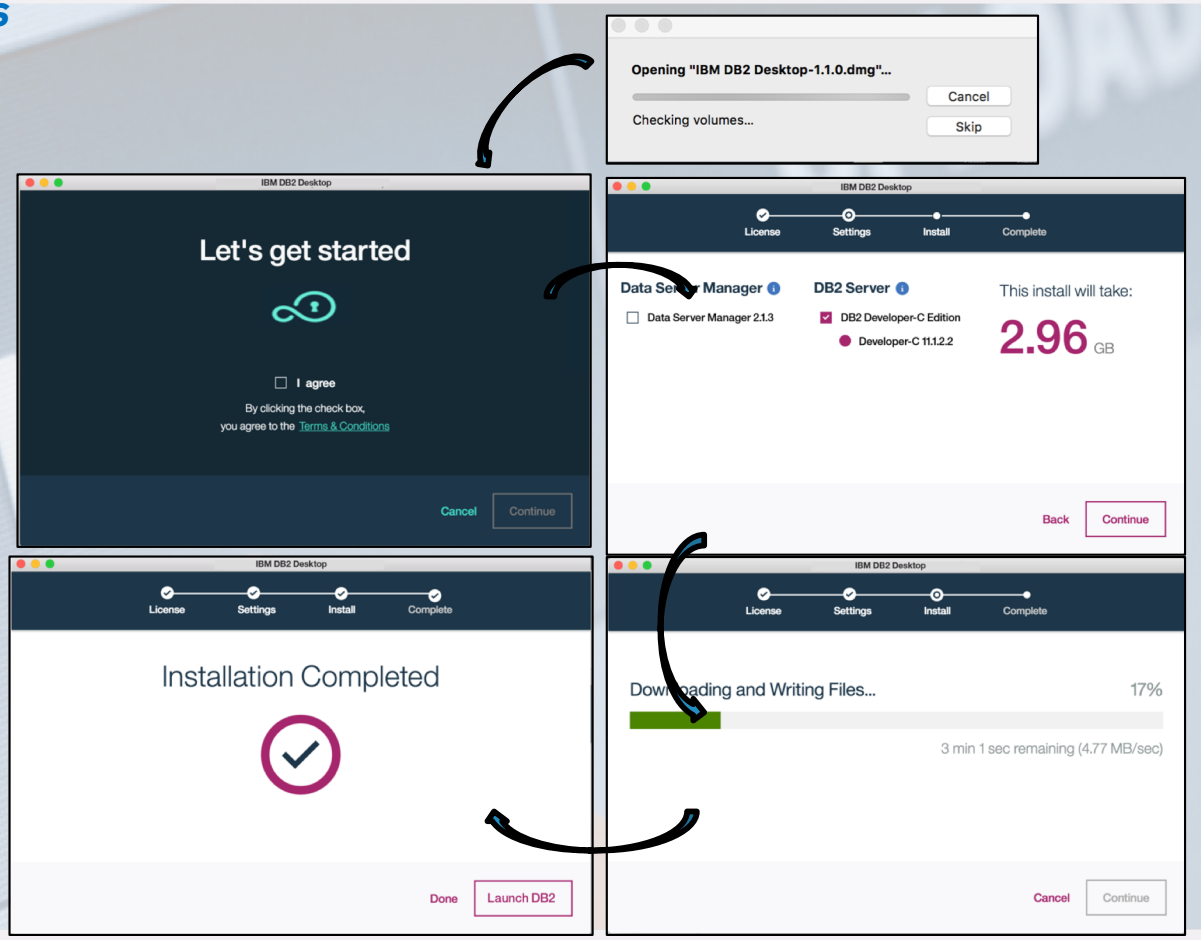
April 4, 2018

Brad Rowen
bjrowen@us.ibm.com

Db2 Developer Community Edition – Optimized for Docker

Up, Running, and Productive in Minutes

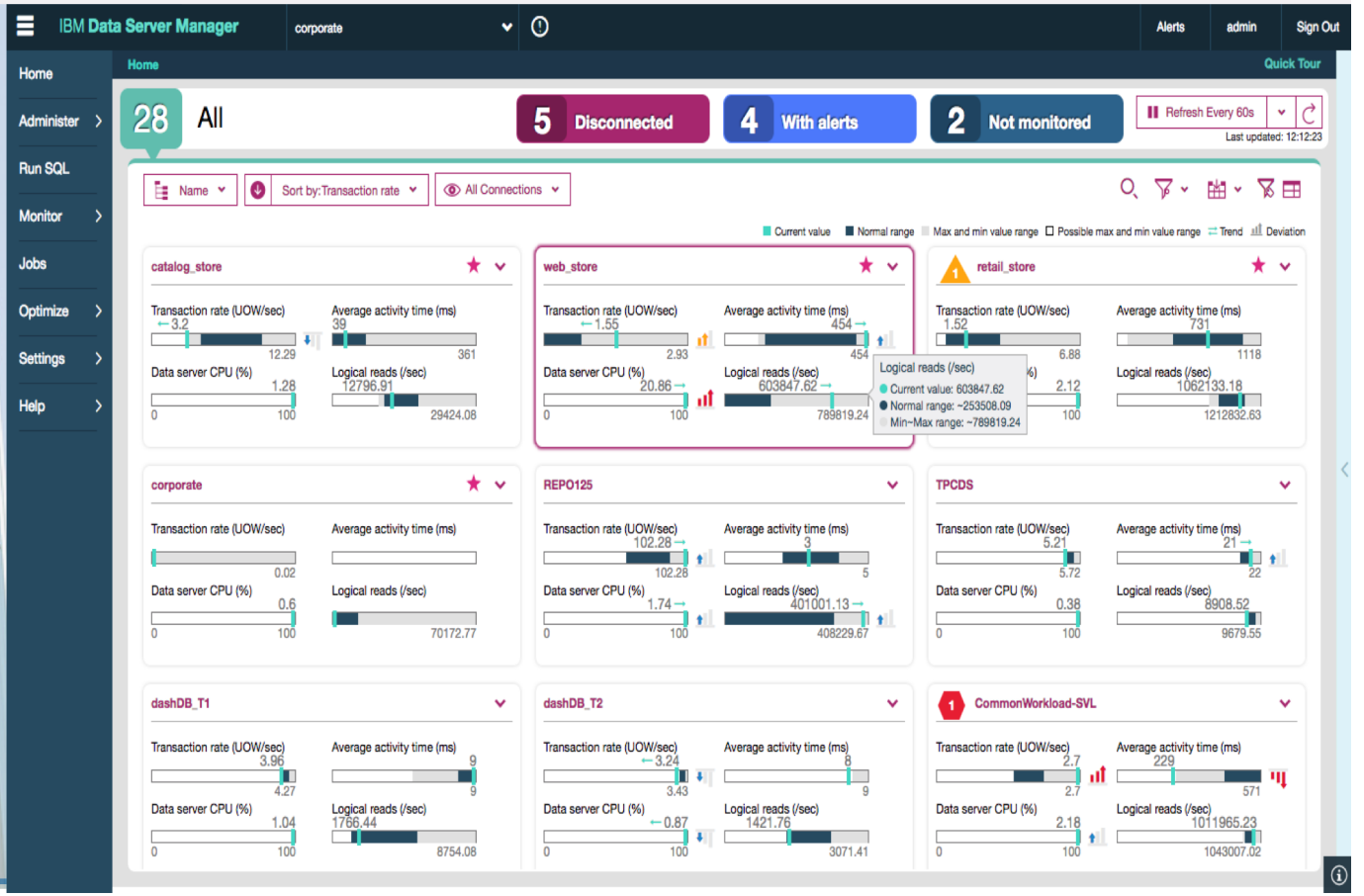
- **Marketplace or Discovery Portal – single point of download**
- **Download platform-specific executable (Mac, Windows, or Linux)**
- **Small download, less than 15 minutes from "I want" to "I got"**
- **3 clicks and userID/pwd input only**



IBM Data Server Manager – In a Docker Container

Monitor, Administer, Manage, and Tune your Environment

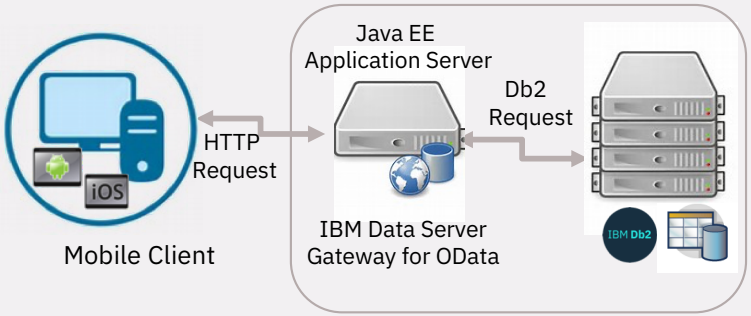
- **A single pane of glass**
- **Simple to install and setup**
- **Scales to 100s of databases**
- **Historical analysis:**
  - minutes, hours, days
- **Smart alerts with recommendations**



Open Data Protocol Gateway

Simplifying Access to Data Sources from Mobile Devices

- **The IBM OData Gateway provides a RESTful service that provides for the following operations:**
  - GET (SELECT)
  - POST (INSERT)
  - PUT (UPDATE a full row)
  - PATCH (UPDATE selected columns in a row)
  - DELETE (Delete a row)

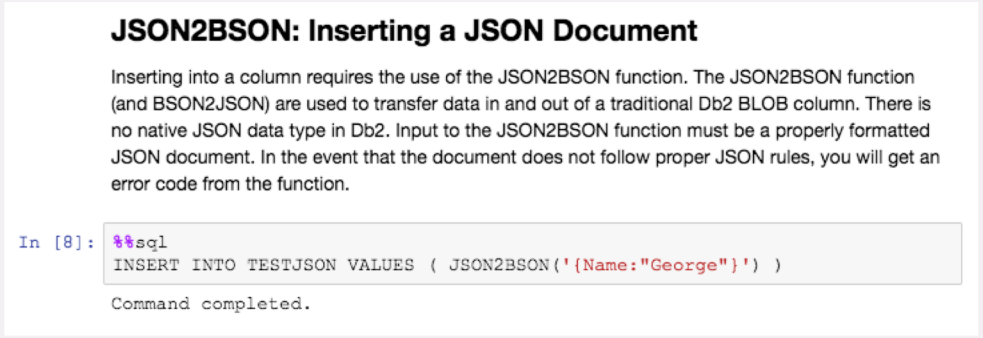



```
url = http://localhost:9080/ODataOne/ODataService/SAMPLE-db2ab6eeb40d4a75a525333ee610fb50

header = {
         'Content-Type':'application/json',
         'Accept':'application/json'
         }

OData = "/EMPLOYEES?$select=LASTNAME,SALARY&$filter=SALARY gt 50000"

RESTful.get(url + OData, headers=header)
```

Jupyter Notebooks and Python

A Modern Approach to Document and Prototype Code

- **Combine SQL, Python, Scala with Documentation - Markdown**
- **Combine SQL with the power of advanced analytics**
- **One line pivot**
- **Simple sort and group**
- **Powerful graphing**
- **Document management processes**
- **Built to share**



JSON2BSON: Inserting a JSON Document

Inserting into a column requires the use of the JSON2BSON function. The JSON2BSON function (and BSON2JSON) are used to transfer data in and out of a traditional Db2 BLOB column. There is no native JSON data type in Db2. Input to the JSON2BSON function must be a properly formatted JSON document. In the event that the document does not follow proper JSON rules, you will get an error code from the function.

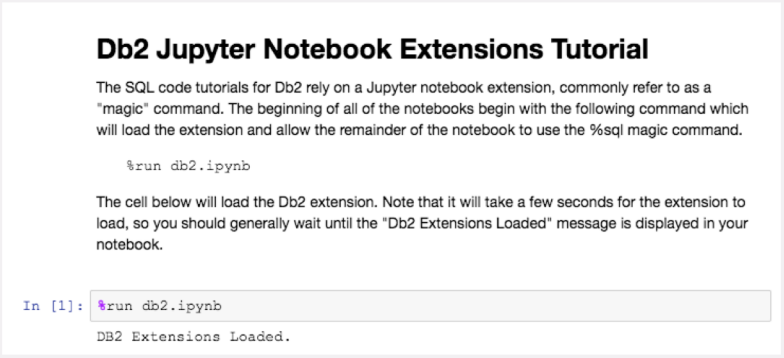
```
In [8]: %%sql
        INSERT INTO TESTJSON VALUES ( JSON2BSON('{Name:"George"}') )
```

```
Command completed.
```

Making Magic with Db2

Extending Jupyter Notebooks to Access Db2

- **Dedicated Db2 Magic Commands**
- **Secure and persistent database connections**
- **One command from SQL to DataFrame**
- **SQL Formatting in Cell**
- **Graphing**



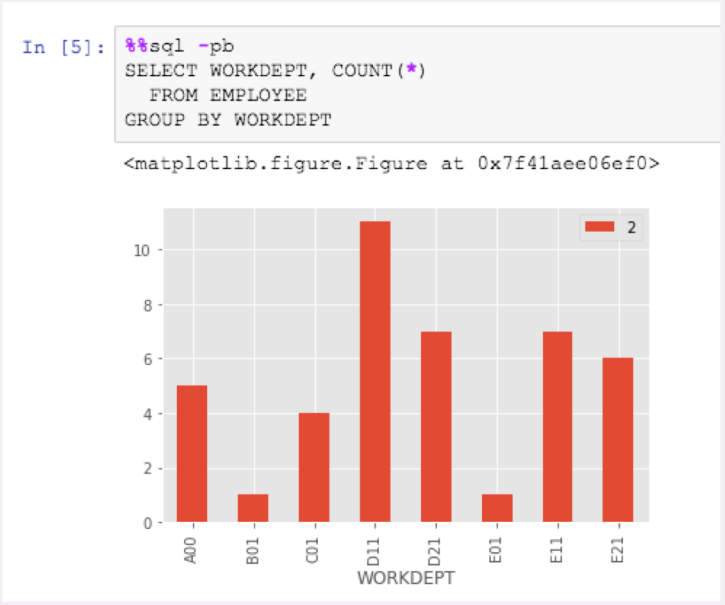
Db2 Jupyter Notebook Extensions Tutorial

The SQL code tutorials for Db2 rely on a Jupyter notebook extension, commonly refer to as a "magic" command. The beginning of all of the notebooks begin with the following command which will load the extension and allow the remainder of the notebook to use the %sql magic command.

```
%run db2.ipynb
```

The cell below will load the Db2 extension. Note that it will take a few seconds for the extension to load, so you should generally wait until the "Db2 Extensions Loaded" message is displayed in your notebook.

```
In [1]: %run db2.ipynb
DB2 Extensions Loaded.
```



```
In [5]: %%sql -pb
SELECT WORKDEPT, COUNT(*)
  FROM EMPLOYEE
GROUP BY WORKDEPT
```

```
<matplotlib.figure.Figure at 0x7f41aee06ef0>
```


github.com/DB2-Samples/db2jupyter

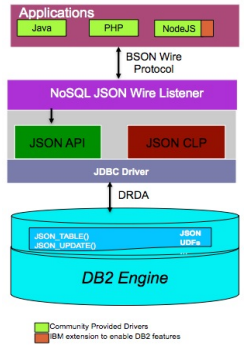
What is JSON?

JSON Explained

NoSQL Wire Listener

JSON support

- **Initial JSON support in Db2 10.5 FP1 consisted of:**
  - IBM NoSQL Wire Listener, JSON Java APIs, JSON command line
- **Db2 11.1.2.2 will add full support for direct SQL access to underlying JSON interfaces used by externals above**
  - Integrated installation/configuration during database creation or migration update
    - No need to catalog JSON functions manually
  - High-level documentation on their existence similar to what Db2 for z/OS already provides
    - JSON2BSON, BSON2JSON are catalogued as part of SYSIBM schema
    - JSON_VAL is function built into the SQL syntax
    - All other JSON functions are part of the SYSTOOLS schema
  - Overview of all functions and how to use them can be found in Db2 11 ebook at: http://ibm.box.com/v/DB2v11eBook



11.1.2.2 SQL functions

Schema	Name	Comments
SYSTOOL	BSON2JSON	Convert BSON formatted document into JSON strings
SYSTOOL	BSON_VALIDATE	Checks to make sure that a BSON field in a BLOB object is in a correct format
SYSTOOL	JSON2BSON	Convert JSON strings into a BSON document format
SYSTOOL	JSON_GET_POS_ARR_INDEX	Find a value within an array
SYSTOOL	JSON_LEN	Returns the count of elements in an array type inside a document
SYSTOOL	JSON_TABLE	Returns a table of values for an array field
SYSTOOL	JSON_TYPE	Returns the data type of a specific field within a JSON document
SYSTOOL	JSON_UPDATE	Update a field or document using set syntax
SYSIBM	JSON_VAL	Extracts data from a JSON document into SQL data types

Legal Disclaimer

- The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
- All references to fictitious companies are used for illustration purposes only.