

# Hybrid Cloud Solutions

Access your data from Everywhere with  
CP4D/Data Fabric & Virtualization



**Benoit MAROLLEAU – Cloud/AI Architect**

Hybrid Cloud Solutions – IBM Client Engineering | EMEA

[benoit.marolleau@fr.ibm.com](mailto:benoit.marolleau@fr.ibm.com)

<https://ibm.biz/bma-wiki>

# RetailOne Use case

- Objectives

How a data fabric can help manipulate and filter data, and expose them to the appropriate target system?

Dispersed data sources :

- Global Product Catalog / Prices / Product classification
- CRM
- E-Commerce : Orders
- ERP per **Store** : Billing / Stocks/ Assortment ...

ETL Simplification with Data consumption - Apache NIFI (ETL) or Salesforce

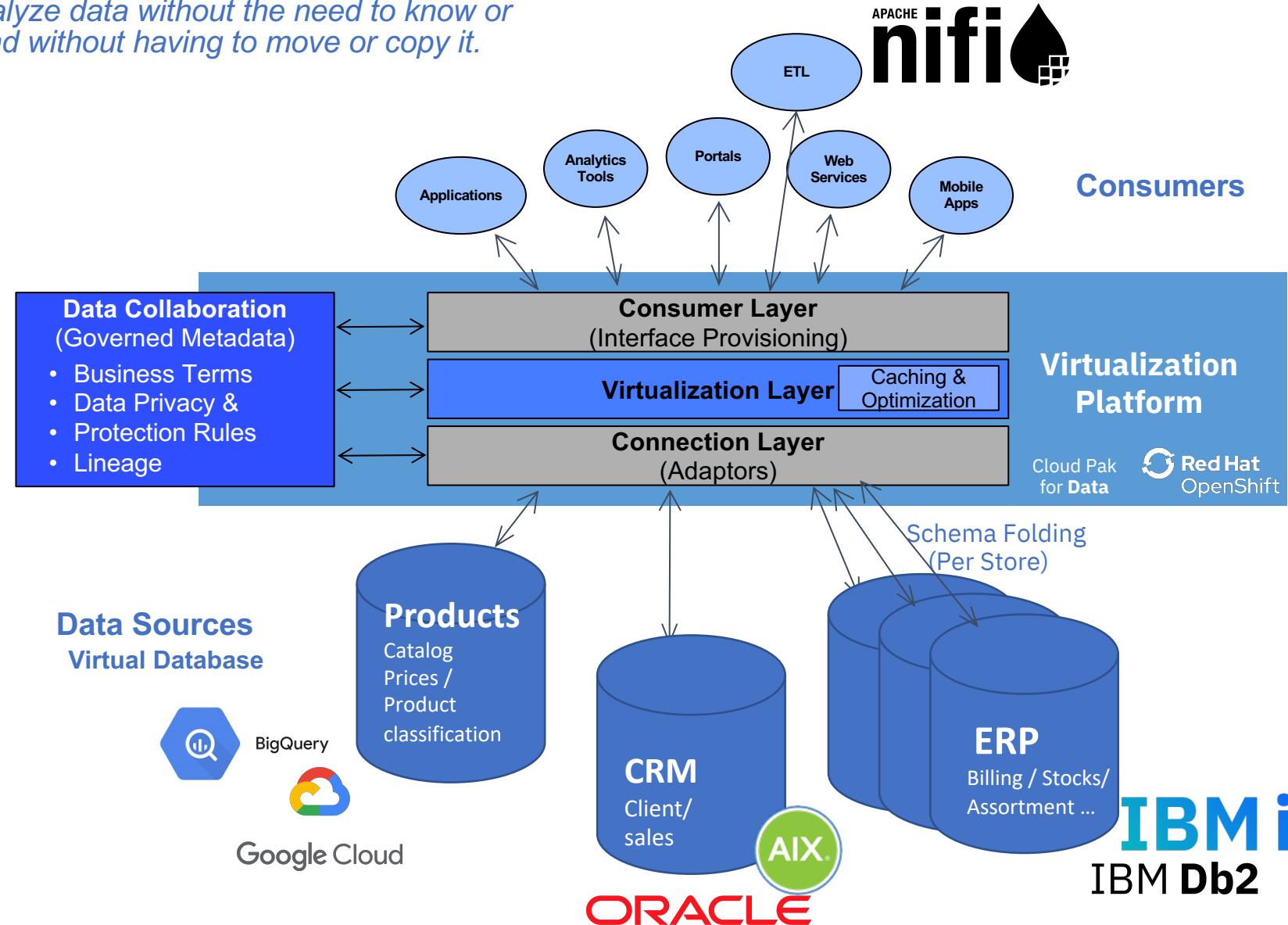
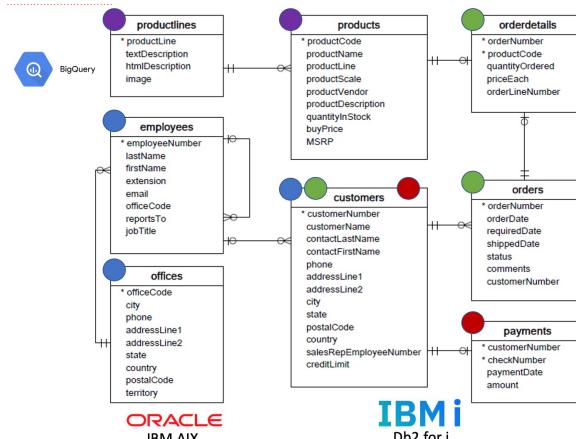
# RetailOne Demo

## IBM Data Fabric pattern: Governed Data Virtualization

*The ability to view, access, manipulate and analyze data without the need to know or understand its physical format or location, and without having to move or copy it.*

### Impact of modern Information Architecture

- Data fabric, on a cloud native architecture
- Reduces data silos and provides a single view of trusted business data
- Reduce data copies and movement
- Enable self-service and data democratization
- Centralized control and governance
- Real-time sharing of data with business users
- Operationalize AI with trust and transparency



# RetailOne Demo

## Integrated Governance

*The ability to classify, mask and protect sensitive data from a single pane of glass*

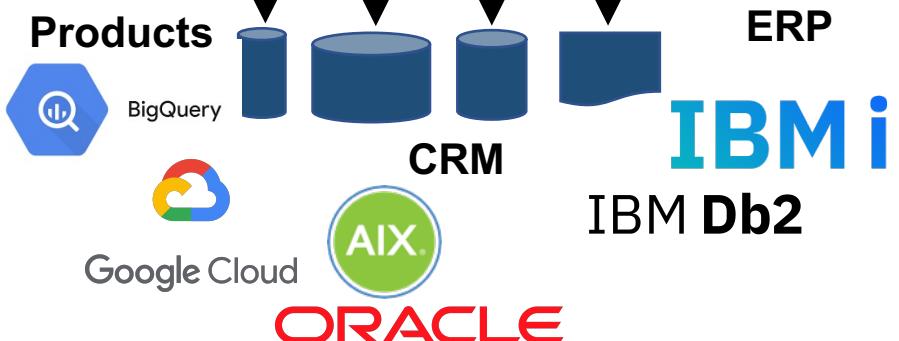
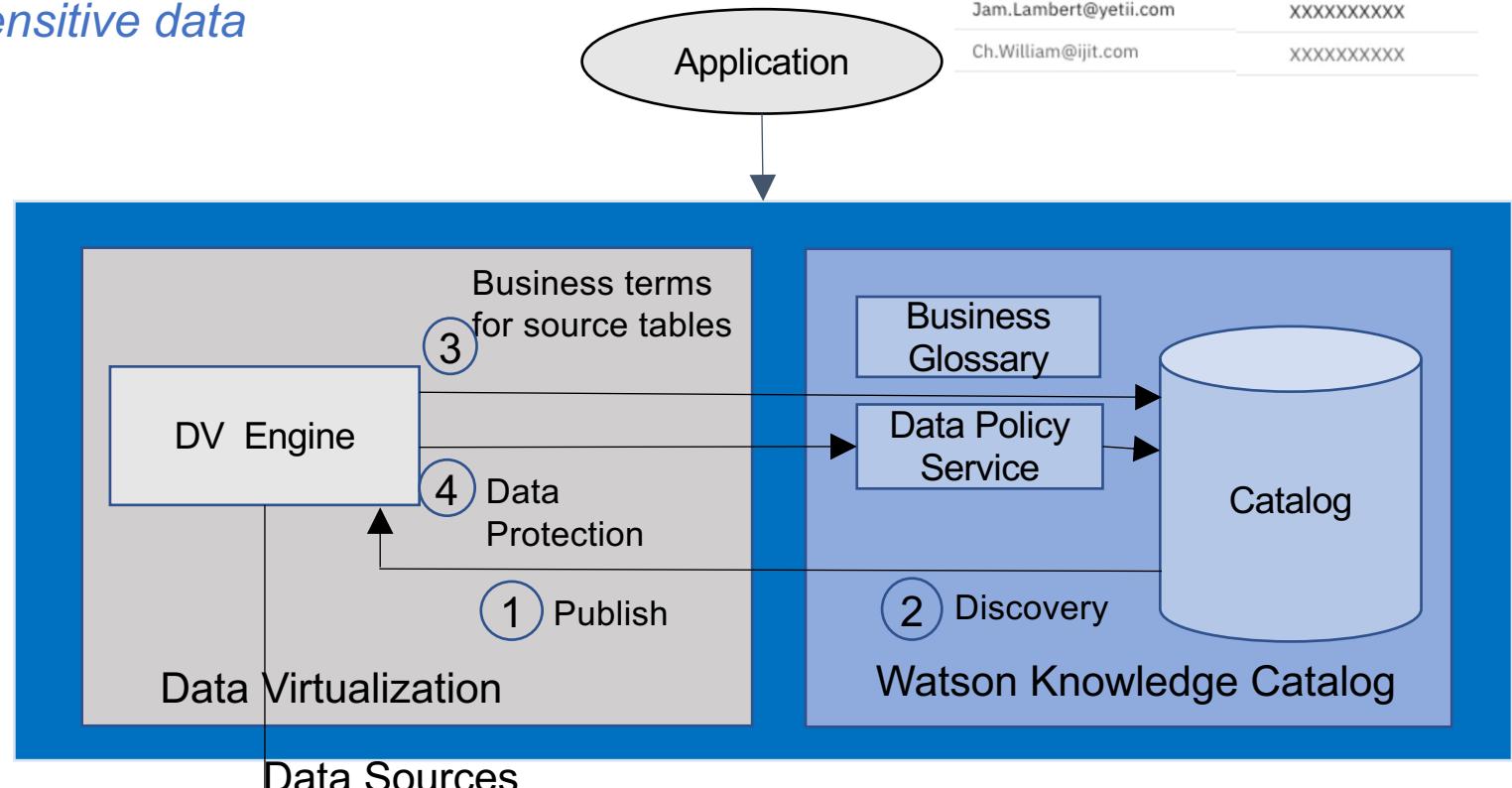
DV Publishes remote assets to WKC.

WKC Discovery performs classification, scoring, term association

DV retrieves Business Terms for the source tables to give the users a common understanding of the data.

DV obtains information about the policies it needs to enforce upon data access.

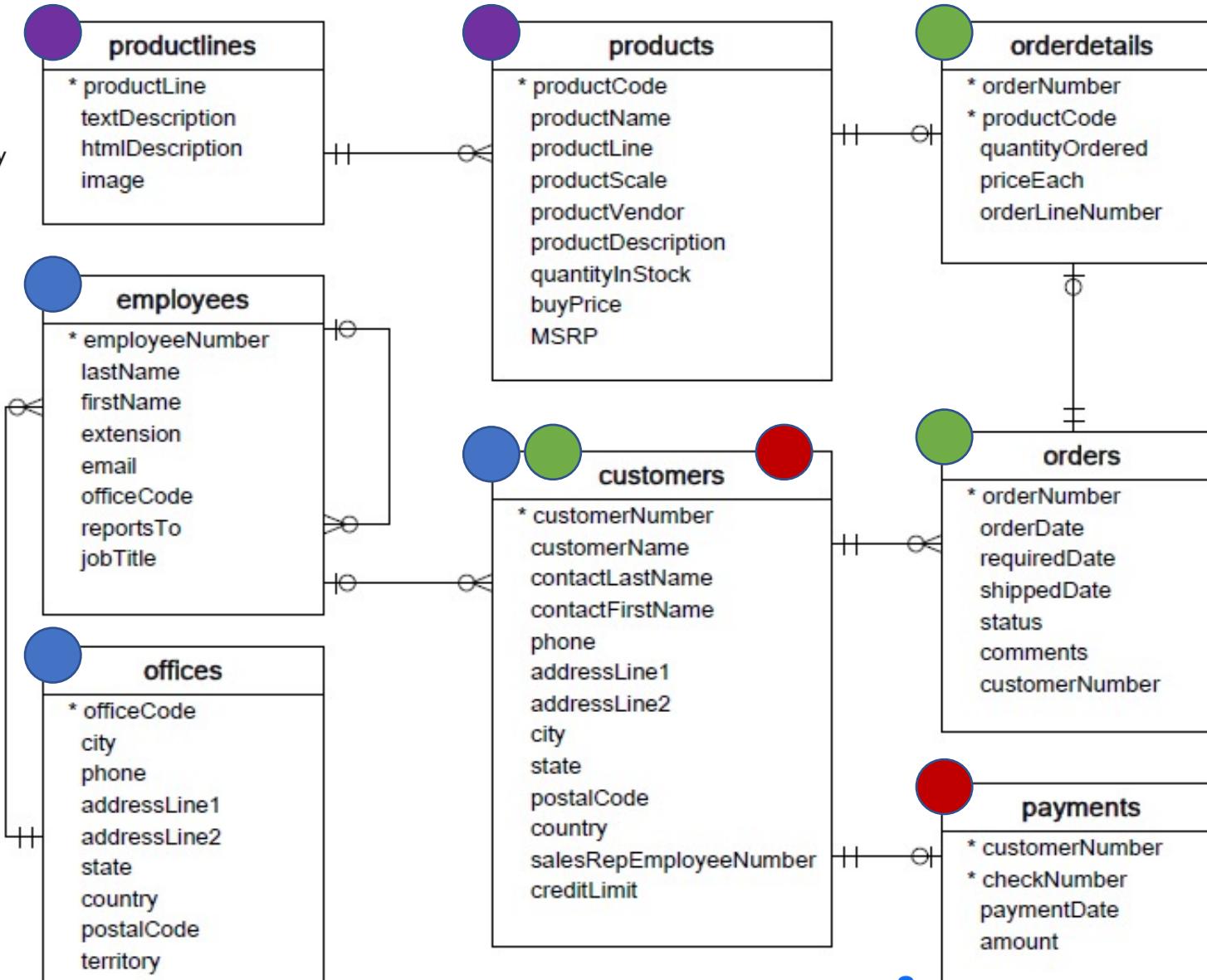
EMAIL VARCHAR	CARD_NO VARCHAR
Eugene.Brock@snailmail.com	XXXXXXXXXX
Su.Chandler@nada.com	XXXXXXXXXX
Et.Merrill@nada.com	XXXXXXXXXX
Na.Wells@snailmail.com	XXXXXXXXXX
Jam.Lambert@yetii.com	XXXXXXXXXX
Ch.William@ijit.com	XXXXXXXXXX



# RetailOne Demo



BigQuery



- Product Catalog / Prices / Product classification
- CRM
- E-Commerce : Orders
- ERP per Store : Billing / Stocks/ Assortment ...

# RetailOne Demo

## IBM Data Fabric pattern: Governed Data Virtualization

**Data consist of the following databases/schemas/tables:**

- Big Query - GCP

- Products: stores a list of scale model cars.
- ProductLines: stores a list of product line categories.



- Db2 for i - On Prems :

- Customers: stores customer's data.
- Orders: stores sales orders placed by customers.
- OrderDetails: stores sales order line items for each sales order.
- Payments: stores payments made by customers based on their accounts.

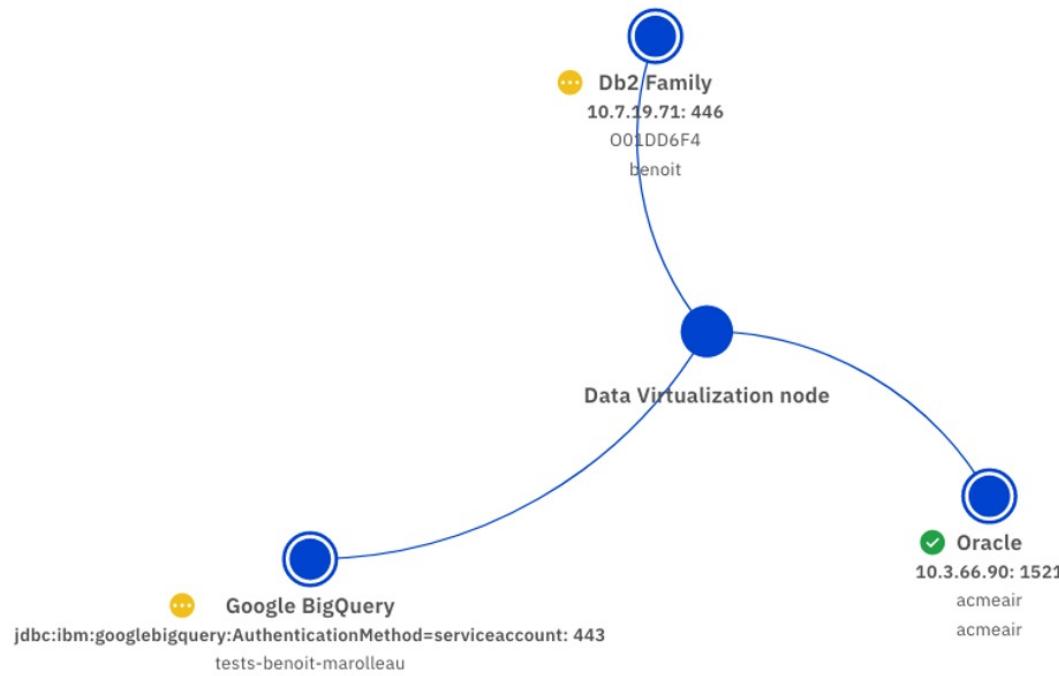


- Oracle AIX - On prems :

- Employees: stores all employee information as well as the organization structure such as who reports to whom.
- Offices: stores sales office data.



# RetailOne Demo



My instances / data-management-console / dv-1637853171439999 /

## Join virtual objects

Back

Click and drag from one table row to the other to create a join key.

Table 1: BOOKING

Find columns		
<input checked="" type="checkbox"/>	Column name	Data type
<input checked="" type="checkbox"/>	ID	VARCHAR
<input checked="" type="checkbox"/>	CUSTOMERID	 VARCHAR
<input checked="" type="checkbox"/>	FLIGHTID	VARCHAR
<input checked="" type="checkbox"/>	FLIGHTSEGMENTID	VARCHAR
<input checked="" type="checkbox"/>	DATEOFPICKUP	TIMESTMP

Table 2: CUSTOMER

Find columns		
<input checked="" type="checkbox"/>	Column name	Data type
<input checked="" type="checkbox"/>	ID	 VARCHAR
<input checked="" type="checkbox"/>	PASSWORD	VARCHAR
<input checked="" type="checkbox"/>	STATUS	VARCHAR
<input checked="" type="checkbox"/>	TOTAL_MILES	INTEGER
<input checked="" type="checkbox"/>	MILES_YTD	INTEGER
<input checked="" type="checkbox"/>	PHONENUMBER	VARCHAR

Run SQL ▾

\* Untitled - 1

Add new script +  

```
1 SELECT * FROM "ACME"."CUSTOMER-BOOKINGS"
```

dv-1637853171439999: Result - Dec 3, 2021 4:31:19 PM  

SELECT \* FROM "ACME"."CUSTOMER-BOOKINGS"

Run time: 2.303 s

Result set 1

ACME_CUSTOMER-Oracle_ID	PASSWORD	STATUS	TOT
uid0@email.com	password	password	12
uid0@email.com	password	password	12

# RetailOne Demo

My instances / data-management-console / dv-1645454458830998 /

Data request [\(i\)](#)     Project [\(i\)](#)     Virtualized data

Publish to catalog [\(i\)](#)

RetailOne ▼

Default Catalog ▼

## Objects to be virtualized

Table	Schema	Source schema	Connection names	Databases/File Path	Hostname: Port	Grouped tables
CUSTOM ...	RETAIL <span style="float: right;">X ▼</span>	RETAILCRM	ERP-Dd2fori-Store1	001DD6F4	10.7.19.71: 446	1
ORDERS	RETAIL <span style="float: right;">X ▼</span>	RETAILCRM	ERP-Dd2fori-Store1	001DD6F4	10.7.19.71: 446	1
ORDERD ...	RETAIL <span style="float: right;">X ▼</span>	RETAILCRM	ERP-Dd2fori-Store1	001DD6F4	10.7.19.71: 446	1
PAYMENTS	RETAIL <span style="float: right;">X ▼</span>	RETAILCRM	ERP-Dd2fori-Store1	001DD6F4	10.7.19.71: 446	1
productli ...	RETAIL <span style="float: right;">X ▼</span>	"tests-benoit-marolleau...	Products-GCP-BigQuery	tests-benoit-marolleau	jdbc:ibm:googlebigque...	1
products	RETAIL <span style="float: right;">X ▼</span>	"tests-benoit-marolleau...	Products-GCP-BigQuery	tests-benoit-marolleau	jdbc:ibm:googlebigque...	1
OFFICES	RETAIL <span style="float: right;">X ▼</span>	ACMEAIR	CRM-OracleAIX	acmeair	10.3.66.90: 1521	1
EMPLOY ...	RETAIL <span style="float: right;">X ▼</span>	ACMEAIR	CRM-OracleAIX	acmeair	10.3.66.90: 1521	1

# RetailOne Demo

Example:

**How to I get my Top 200 Sales from multiple data sources in one simple query ?**

Simplified & secured access to business data

- Virtual Views pointing to multiple dispersed joined databases
- Up to date data (no data copy)
- Reinforced Data Governance: Catalog, Masking, Access Management

# With DV, you can join 2 virtual tables together or create your own virtual views:

```
CREATE VIEW "RETAIL".TOPSALES AS
  SELECT cus.CUSTOMERNAME AS "CustomerName (Db2 for i)",
         prod.productName AS "Product (GCP Big Query)",
         pay.paymentDate AS "Sales Date (Db2 for i)",
         emp.lastname AS "Sales Rep Name (Oracle)",
         O.CITY AS "Sales Rep City(Oracle)",
         det.quantityOrdered AS "Qty(Db2 for i)",
         det.priceEach AS "Unit Price",
         det.quantityOrdered * det.priceEach AS "TOTAL"

  FROM "RETAIL"."CUSTOMERS" cus,
       "RETAIL"."products" prod,
       "RETAIL"."PAYMENTS" pay,
       "RETAIL"."EMPLOYEES" emp,
       "RETAIL"."ORDERS" orders,
       "RETAIL"."ORDERDETAILS" det ,
       "RETAIL"."OFFICES" O

  WHERE cus.customerNumber=pay.customerNumber
    AND cus.SALESREPEMPLOYEENUMBER=emp.employeeNumber
    AND orders.orderNumber=det.orderNumber
    AND det.productCode= prod.productCode"
    AND emp.officecode = O.officecode);
```

```
%sql SELECT * from "RETAIL"."TOPSALES" ORDER BY TOTAL DESC FETCH FIRST 200 ROWS ONLY;
```

## ## Making Complex SQL Simple to Consume

In [ ]: # With DV, you can join 2 virtual tables together or create your own virtual views:

In [51]: %sql SELECT \* from "RETAIL"."TOPSALES" ORDER BY TOTAL DESC FETCH FIRST 20 ROWS ONLY;

Out[51]:

	CustomerName (Db2 for i)	Product (GCP Big Query)	Sales Date (Db2 for i)	Sales Rep Name (Oracle)	Sales Rep City(Oracle)	Qty(Db2 for i)	Unit Price	TOTAL
0	Souveniers And Things Co.	2003 Harley-Davidson Eagle Drag Bike	2004-08-02	Fixter	Sydney	66.0	174.29	11503.14
1	Mini Gifts Distributors Ltd.	2003 Harley-Davidson Eagle Drag Bike	2004-08-28	Jennings	San Francisco	66.0	174.29	11503.14
2	Mini Gifts Distributors Ltd.	2003 Harley-Davidson Eagle Drag Bike	2005-03-05	Jennings	San Francisco	66.0	174.29	11503.14
3	Baane Mini Imports	2003 Harley-Davidson Eagle Drag Bike	2004-11-28	Jones	London	66.0	174.29	11503.14
4	Baane Mini Imports	2003 Harley-Davidson Eagle Drag Bike	2004-11-04	Jones	London	66.0	174.29	11503.14
...	...	...	...	...	...	...	...	...
15	Signal Gift Stores	2003 Harley-Davidson Eagle Drag Bike	2003-06-06	Thompson	San Francisco	66.0	174.29	11503.14
16	Signal Gift Stores	2003 Harley-Davidson Eagle Drag Bike	2004-12-17	Thompson	San Francisco	66.0	174.29	11503.14

# RetailOne Demo

My instances / dv-1645454458830998 / Virtualization /

✓ Generating recommendations...: This table is automatically refreshed when the process is completed. Please check back later.

✓ Success: Started recommendation cycle

Data caches

Queries

## Cache storage

Available

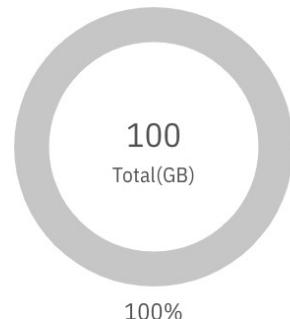
**100.0 GB**

Inactive data caches

**0.0 KB**

Active data caches

**0.0 KB**



- Available
- Inactive data caches
- Active data caches

## Responsiveness

Total queries

**4**

Avg response time

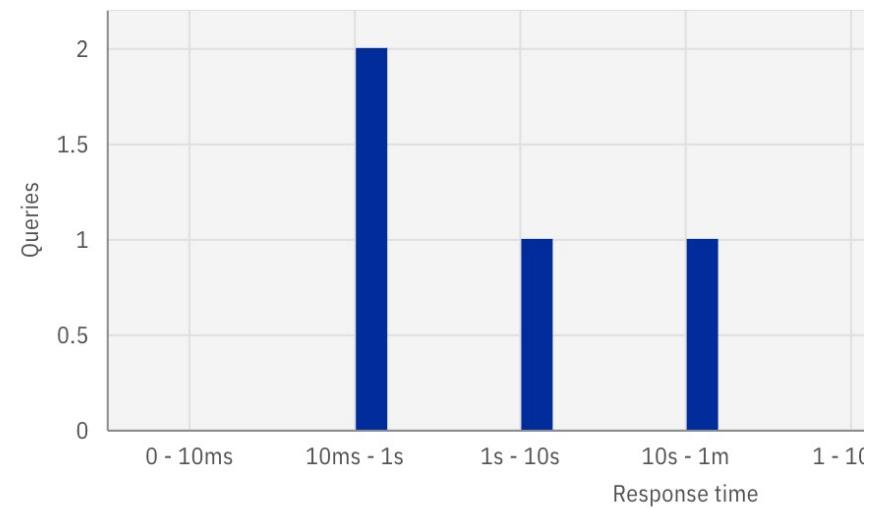
**3s 880ms**

Using cache

**0.0%**

Not using cache

**100.0%**



# RetailOne Demo

IBM Cloud Pak for Data    All    Search

Catalogs / Default Catalog

Data

## RETAIL.CUSTOMERS

Overview Asset Access

Current profile Last profile Columns  
166 classifiers 28 Feb 2022 - 3:15 pm View log 13

AME CONTACTLASTNAME Type: Varchar

Frequency

Name	Count
Young	100%
Frick	0%
Brown	100%
Nelson	0%
Hernandez	100%
King	0%
Murphy	100%
Thompson	0%
Franco	100%
Taylor	0%

Cancel

Edit data class

Search: phone

All data classes

- Not classified 61% matches
- US Phone Number 39% matches
- PhoneNumber -

Remove Add to project

Delete profile Update profile

ADDRESSLINE1 Type: Varchar

RETAIL.CUSTOMERS

Overview Asset Access Review Profile Activities

Current profile Last profile Columns Rows  
166 classifiers 28 Feb 2022 - 3:15 pm View log 13 122

Legend: Matches (green), Mismatches (purple), Missing (black)

AME CONTACTLASTNAME Type: Varchar

Frequency

Name	Count
Young	100%
Frick	0%
Brown	100%
Nelson	0%
Hernandez	100%
King	0%
Murphy	100%
Thompson	0%
Franco	100%
Taylor	0%

CONTACTFIRSTNAME Type: Varchar

Frequency

Name	Count
Julie	100%
Michael	0%
Sue	100%
Mike	0%
Leslie	100%
Valarie	0%

PHONE Type: Varchar

Frequency

Number	Count
6175558555	100%
+358 9 8045 555	0%
4155551450	100%
+64 9 5555500	0%
91244555	100%
6562-9555	0%

ADDRESSLINE1 Type: Varchar

Frequency

Address	Count
Drammensveien 126A	100%
Kingsfordweg 151	0%
11328 Douglas Av.	100%
Rte des Arsenaux 41	0%
Via Ludovico il Moro 22	100%
Berliner Platz 43	0%

Remove Add to project +

Delete profile Update profile

# RetailOne Demo

IBM Cloud Pak for Data    All    Search

## New data protection rule

**Rule Details**

Name: Personal Sensitive Information Rule

Type: Access

Business definition: RetailOne Personal Sensitive Information Definition

**Condition 1**

If: Data class contains any  
PhoneNumber

Add new condition +

**Action**

then: mask data in columns containing Data class  
PhoneNumber

Select how to mask data:

**Redact** (selected): Before 452-821-1120, Replace data with Xs

**Substitute**: Before 452-821-1120, Replace data with values that don't match the original format

**Obfuscate**: Before 452-821-1120, Replace data with similarly formatted values

# RetailOne Demo

```
%sql SELECT * from "RETAIL"."CUSTOMERS" ;
```

CUSTOMERNUMBER	CUSTOMERNAME	CONTACTLASTNAME	CONTACTFIRSTNAME	PHONE	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	COUNTRY	SALESREPLOYEEENUM
0	103.0	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale	addr	Nantes	None	44000	France
1	112.0	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.	None	Las Vegas	NV	83030	USA
2	114.0	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road	Level 3	Melbourne	Victoria	3004	Australia
3	119.0	La Rochelle Gifts	Labrune	Janine	40.67.8555	Rue des Graviers, 56 Monte Ottawa	None	Nantes	None	44000	France
4	121.0	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78	None	Stavern	None	4110	Norway

```
%sql SELECT * from "RETAIL"."CUSTOMERS"
```

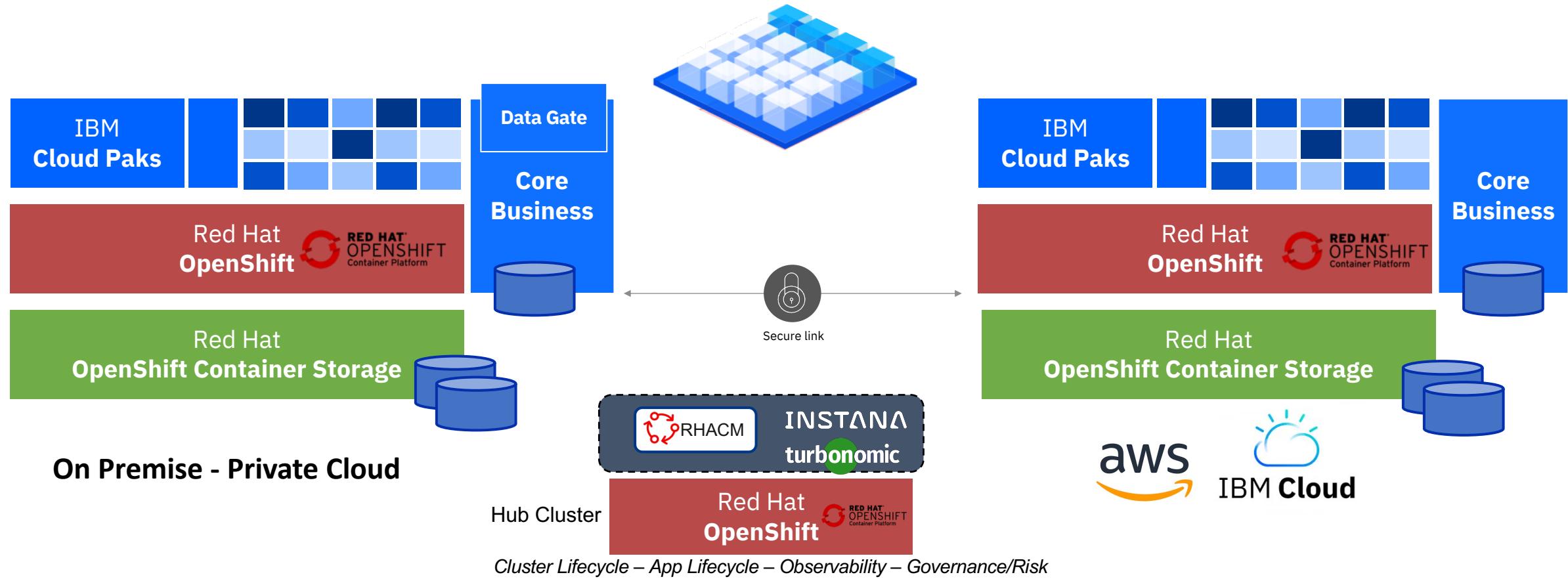
3] :

# Hybrid Cloud Solutions



ANY DATA, ANY CLOUD, ANYWHERE

## IBMs data fabric based on IBM Cloud Pak for Data



# Extend and Standardize your Data Estate

**Connect** heterogeneous data sources

**Join** data using advanced SQL

**Create** views to simplify access

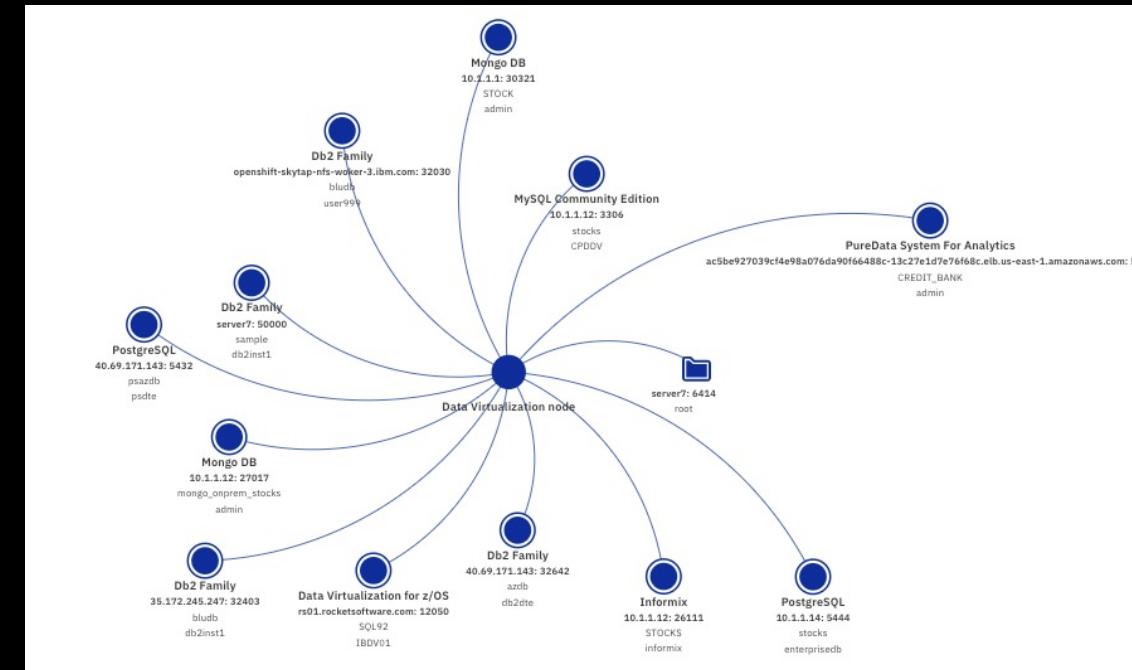
**Protect** data with deep data governance

**Accelerate** performance with local caching

**Connect** with your existing tools

**Build** like a Data Scientist with Watson Studio

**Access** through RESTful services and endpoints



Action \* ⓘ

then mask data in columns containing

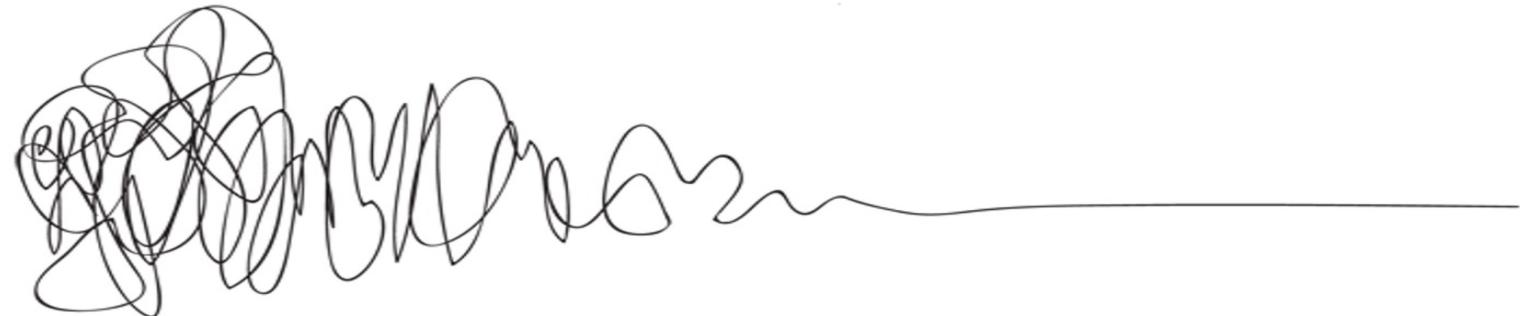
Credit Card Number X

Select how to mask data:

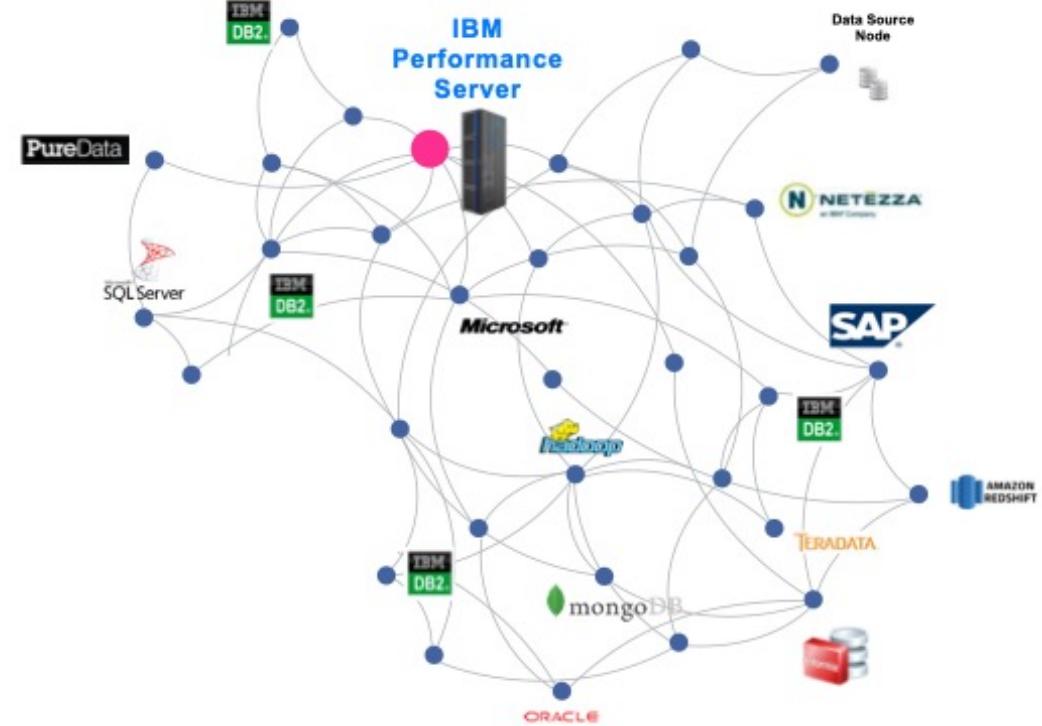
- Redact**: Before 452-821-1120. Replace data with Xs.
- Substitute**: Before 452-821-1120. Replace data with values that don't match the original format.
- Obfuscate**: Before 452-821-1120. Replace data with similarly formatted values.

# Why is it **harder than it looks?**

- Data is in multiple **hetrogeneous** data servers
- Data servers are running **on premises**...
  - ...and on multiple **cloud** providers
- Data is maintained by **different teams**
- **Data governance** is mandatory
- Replicated data is always **old data**



# You can make this

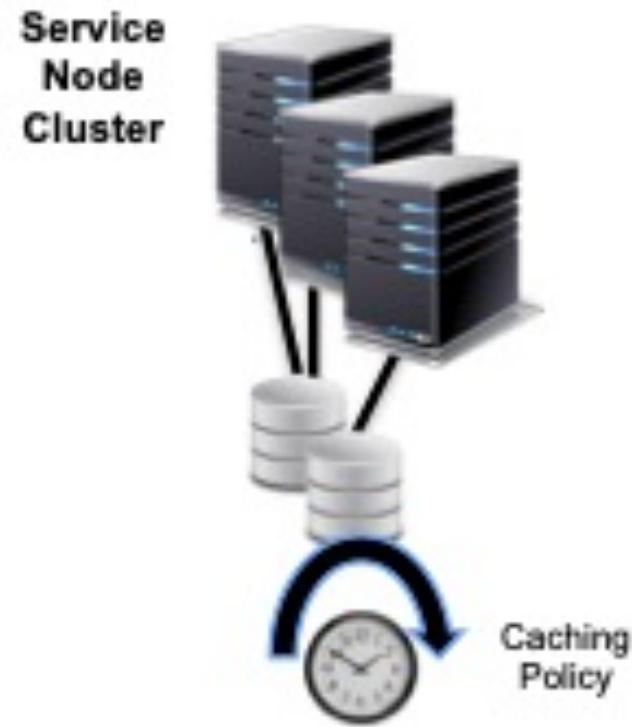


Look like  
**this**

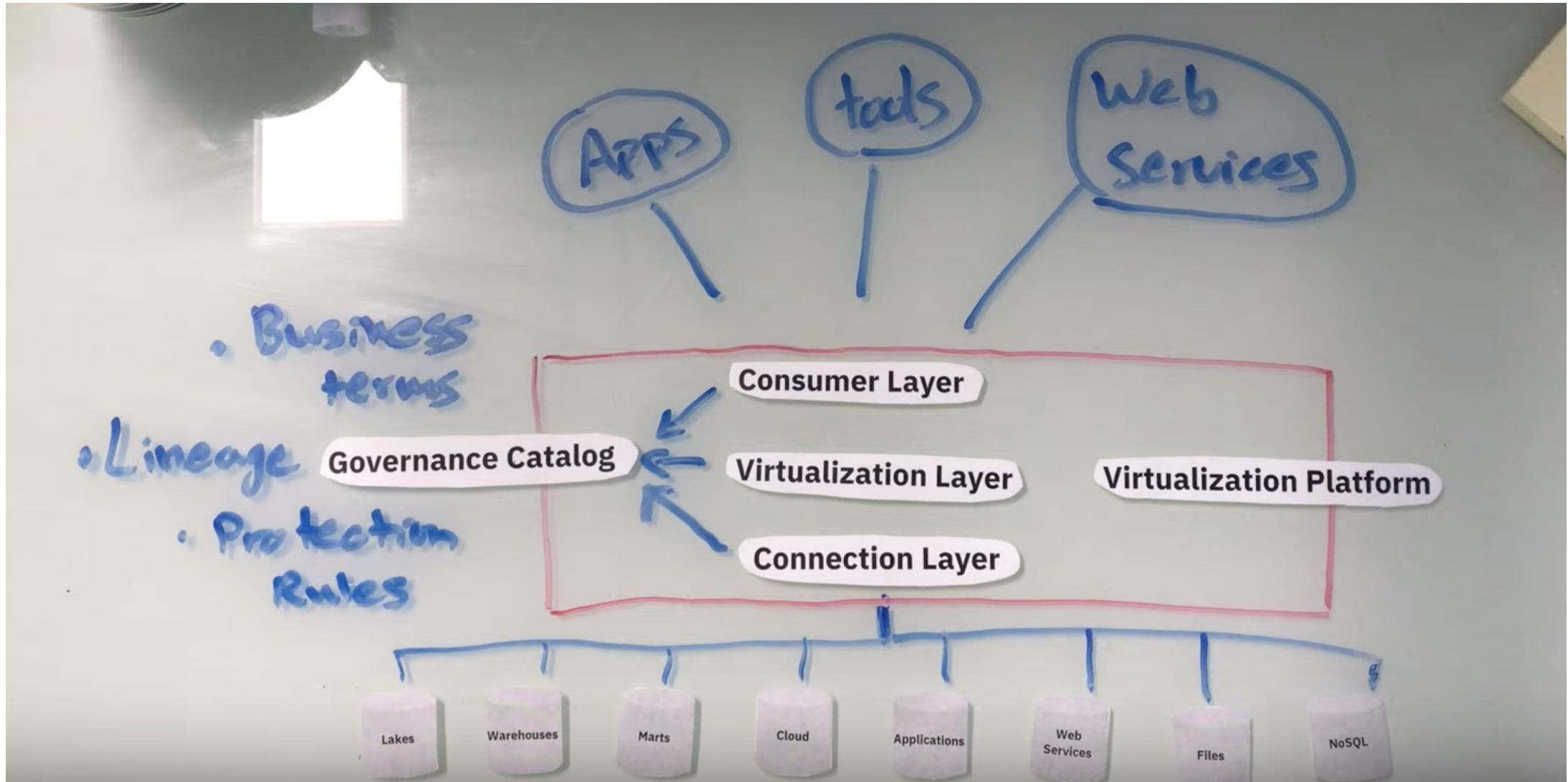


# With this

- IBM Cloud Pak for Data
- **Data Virtualization**

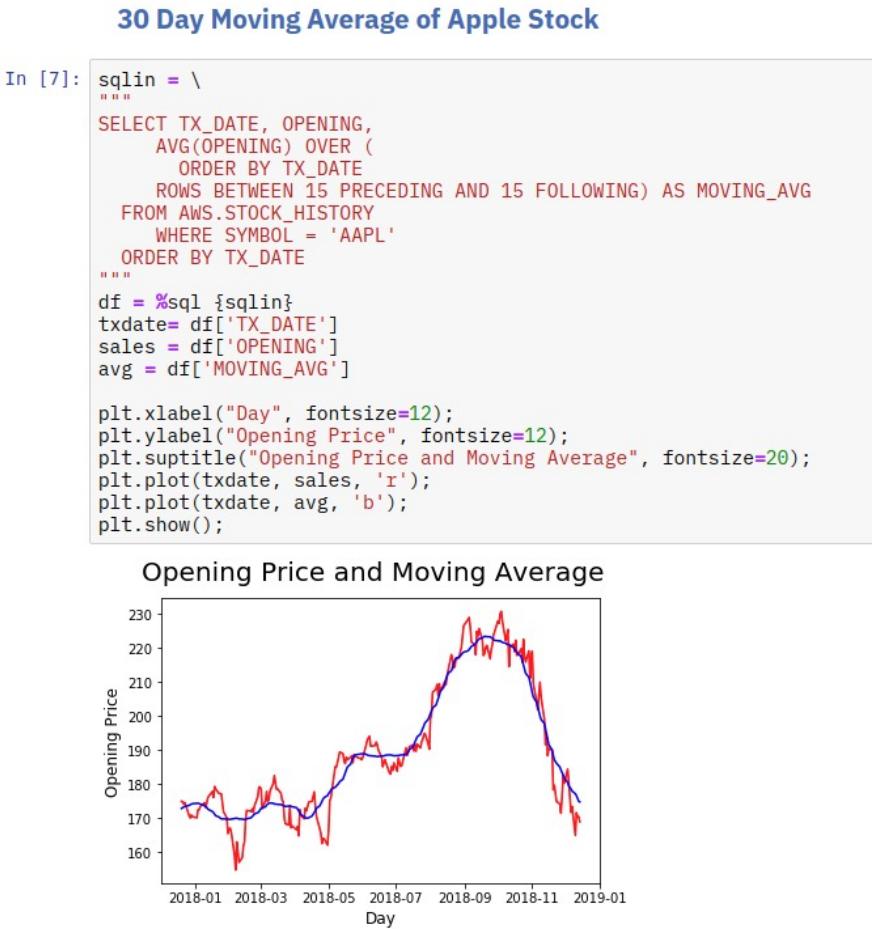


# Data Virtualization



# All your Data Analysis from one Data Source

## One connection for simplicity and performance



# Virtualize Data

Make a remote table work like a local table without moving data

## ④ Data virtualization

Menu ▾ | Virtualize

Browse for:

Tables

Files

### Filters

#### Databases



Db2 Family (219)



z/OS data source (non-Db...)



PostgreSQL (68)



Mongo DB (24)

### Available tables

399 tables

Find tables by name, schema, or column



-	Table	Schemas	Databases
<input type="checkbox"/>	WEBINFO	DB2INST1	INSURE
<input type="checkbox"/>	CUSTOMERS_JSON	AWS	BLUDB
<input checked="" type="checkbox"/>	PORTFOLIO	AWS	BLUDB
<input type="checkbox"/>	STOCK_SYMBOLS	AWS,DB2INST1,DVDEMO,STOCKS,S...	BLUDB,BLUDB,BLUDB,!'

# Join virtual tables

## Make two virtual tables work like a single local table

Menu ▾ | My virtualized data

STOCK X Q

Table	Schema	Created on
STOCK_TRANSACTIONS	AZMONGO	Mar 6, 2020 5:58:46 PM
STOCK_HISTORY	AZMONGO	Mar 6, 2020 5:58:23 PM
STOCK_SYMBOLS	AZMONGO	Mar 6, 2020 5:58:19 PM

### Join virtual objects

Join two columns by selecting a column from one table and then dragging your cursor to a column in the other table.

Table 1: STOCK\_HISTORY

Column Name	Data Type
CLOSE	DOUBLE
HIGH	DOUBLE
LOW	DOUBLE
OPEN	DOUBLE
SYMBOL	VARCHAR
TX_DATE	DATE
VOLUME	INTEGER
_ID	VARCHAR

Table 2: STOCK\_SYMBOLS

Column Name	Data Type
COMPANY	VARCHAR
SYMBOL	VARCHAR
_ID	VARCHAR



Join two virtual tables Open in SQL editor

Join Keys Filters

STOCK_HISTORY	STOCK_SYMBOLS
VAR SYMBOL	VAR SYMBOL

Cancel Preview Join

# Build Views from Complex Queries

Design your own queries so they work like a local table

⟲ | SQL editor

\* Untitled - 1



```
1 CREATE VIEW VIEW_NAME
2     AS
3     SELECT
4         "AZMONGO"."STOCK_HISTORY"."CLOSE" AS "AZMONGO_STOCK_HISTORY_CLOSE",
5         "AZMONGO"."STOCK_HISTORY"."HIGH" AS "AZMONGO_STOCK_HISTORY_HIGH",
6         "AZMONGO"."STOCK_HISTORY"."LOW" AS "AZMONGO_STOCK_HISTORY_LOW",
7         "AZMONGO"."STOCK_HISTORY"."OPEN" AS "AZMONGO_STOCK_HISTORY_OPEN",
8         "AZMONGO"."STOCK_HISTORY"."SYMBOL" AS "AZMONGO_STOCK_HISTORY_SYMBOL",
9         "AZMONGO"."STOCK_HISTORY"."TX_DATE" AS "AZMONGO_STOCK_HISTORY_TX_DATE",
10        "AZMONGO"."STOCK_HISTORY"."VOLUME" AS "AZMONGO_STOCK_HISTORY_VOLUME",
11        "AZMONGO"."STOCK_HISTORY"."_ID" AS "AZMONGO_STOCK_HISTORY__ID",
12        "AZMONGO"."STOCK_SYMBOLS"."COMPANY" AS "AZMONGO_STOCK_SYMBOLS_COMPANY",
13        "AZMONGO"."STOCK_SYMBOLS"."_ID" AS "AZMONGO_STOCK_SYMBOLS__ID"
14    FROM "AZMONGO"."STOCK_HISTORY", "AZMONGO"."STOCK_SYMBOLS"
15    WHERE "AZMONGO"."STOCK_HISTORY"."SYMBOL"="AZMONGO"."STOCK_SYMBOLS"."SYMBOL"
```

# Protect Data using Deep Data Governance

Classify data and protect using active masking

Action \* ⓘ

then mask data in columns containing

Credit Card Number X

Select how to mask data:

- Redact**   
Before 452-821-1120  
Replace data with Xs
- Substitute**   
Before 452-821-1120  
Replace data with values that don't match the original format
- Obfuscate**   
Before 452-821-1120  
Replace data with similarly formatted values

My virtualized data / CUSTOMER\_PAYMENT

**CUSTOMER\_PAYMENT**  
Created on: Oct 5, 2020 11:58:01 AM

Preview Table structure Metadata

3 Columns | 1 Column masked ⓘ  
Preview: 20 rows 1 column is redacted.

CUSTOMER_ID	CARD_TYPE	CARD_NO
5F734E89F0B67E32283DC32F	MCCD	XXXXXXXXXX
5F734E89F0B67E32283DC32E	DYNY	XXXXXXXXXX
5F734F54F0B67E32283DCDBE	VASA	XXXXXXXXXX
5F734F54F0B67E32283DCDC1	VASA	XXXXXXXXXX

In [4]: %sql SELECT \* from STOCK.CUSTOMER\_PAYMENT FETCH FIRST 5 ROWS ONLY

Out[4]:

	CUSTOMER_ID	CARD_TYPE	CARD_NO
0	5E67E30C06A12ADB5AF39A01	PKUP	XXXXXXXXXX
1	5E67E30C06A12ADB5AF39A02	VASA	XXXXXXXXXX
2	5E67E30C06A12ADB5AF39A03	VASA	XXXXXXXXXX
3	5E67E30C06A12ADB5AF39A04	CASH	XXXXXXXXXX
4	5E67E30C06A12ADB5AF39A05	PKUP	XXXXXXXXXX

# Optimize

## Cache results for optimum performance

Menu ▾ | Cache management

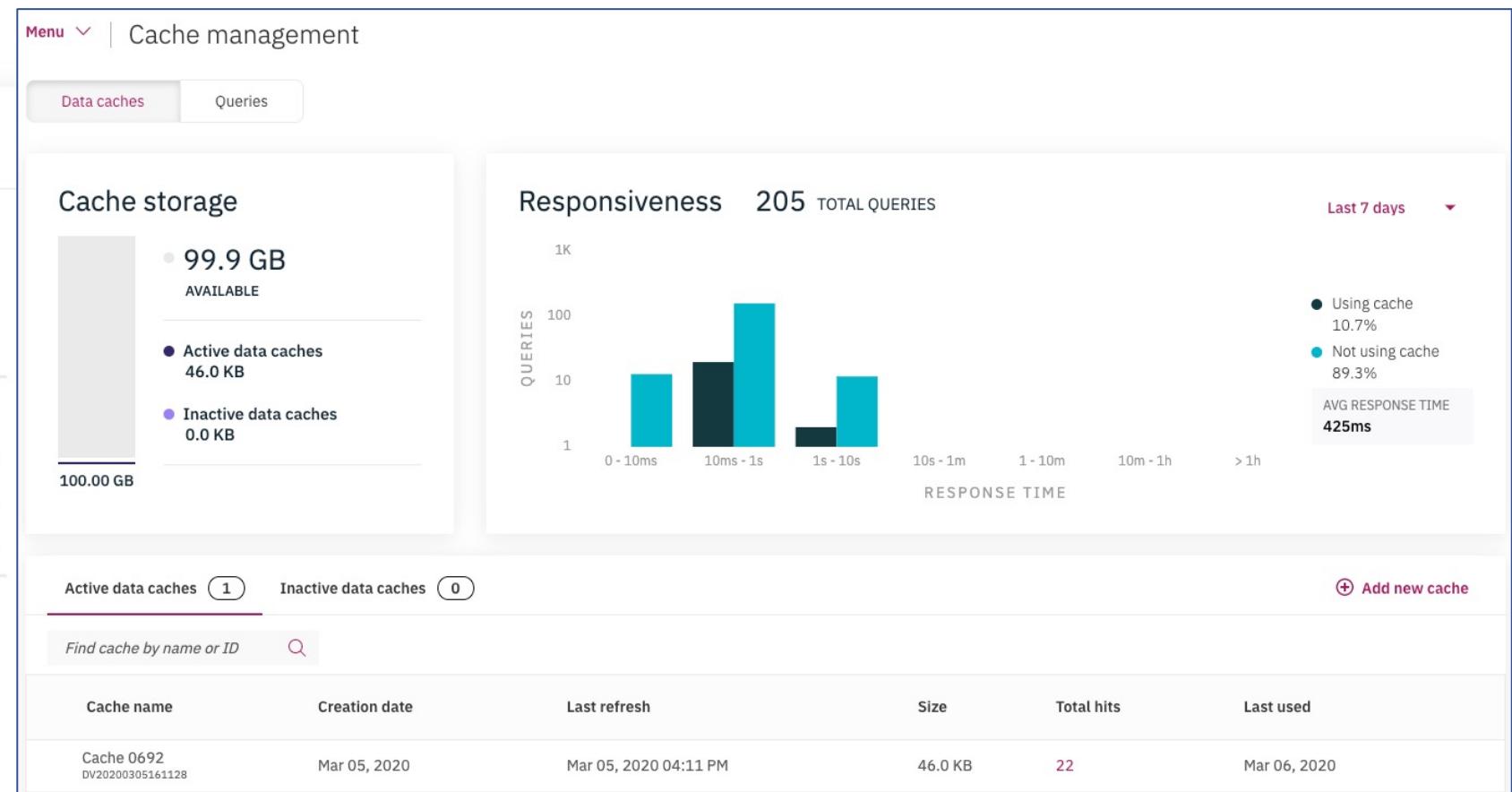
Data caches    Queries

Find by Query ID or Creator  View

Query ID 5318580055266665595

2 Total query executions in the last 7 days

Users	USER999
Avg response time without cache	N/A
Avg response time with cache	0.16 sec
Executions using cache	2



# Connect your Analytics Applications

## Connection configuration resources

With SSL  Without SSL

**Host name:** services-uscentral.skytap.com

**Port number:** 31777

**Database name:** bigsql

**User ID:** user999

**Version:** Compatible with Db2, Version 11.1.0 or later

**Instance ID:** a3f9b9a1-bedd-4862-9152-bc7d50f7c7f6

 [Download SSL Certificate](#)

### JDBC string

```
jdbc:db2://services-uscentral.skytap.com:31777/bigsql:user=user999;password=<your_p
```

 [Copy JDBC String](#)

### More information

 [Db2 driver package \(IBM Knowledge Center\)](#)

 [IBM Data Server Client Packages](#)

 [Connecting CLPPlus to a Db2 database \(IBM Knowledge Center\)](#)

# Connect with Jupyter Hub

## Use advanced Data Scientist Tools

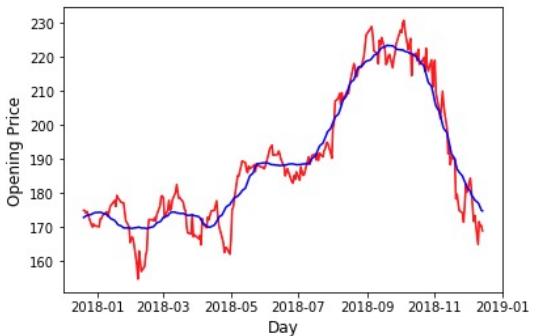
30 Day Moving Average of Apple Stock

```
In [7]: sqlin = \
"""
SELECT TX_DATE, OPENING,
       AVG(OPENING) OVER (
           ORDER BY TX_DATE
           ROWS BETWEEN 15 PRECEDING AND 15 FOLLOWING) AS MOVING_AVG
  FROM AWS STOCK_HISTORY
 WHERE SYMBOL = 'AAPL'
 ORDER BY TX_DATE
"""

df = %sql {sqlin}
txdate= df['TX_DATE']
sales = df['OPENING']
avg = df['MOVING_AVG']

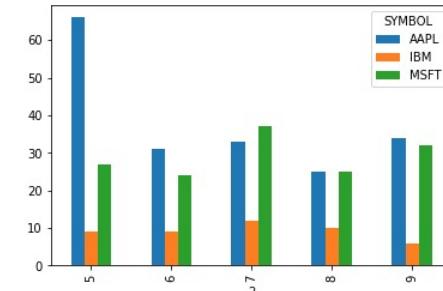
plt.xlabel("Day", fontsize=12);
plt.ylabel("Opening Price", fontsize=12);
plt.suptitle("Opening Price and Moving Average", fontsize=20);
plt.plot(txdate, sales, 'r');
plt.plot(txdate, avg, 'b');
plt.show();
```

Opening Price and Moving Average



Trade volume of IBM versus MSFT and AAPL in first week of November

```
In [8]: %%sql -pb
SELECT SYMBOL, DAY(TX_DATE), VOLUME/1000000 FROM AWS STOCK_HISTORY
WHERE SYMBOL IN ('IBM', 'MSFT', 'AAPL') AND WEEK(TX_DATE) = 45
ORDER BY DAY(TX_DATE) ASC
```



Bought/Sold Amounts of Top 5 stocks

```
In [9]: %%sql
WITH BOUGHT(SYMBOL, AMOUNT) AS
(
  SELECT SYMBOL, SUM(QUANTITY) FROM AWS STOCK_TRANSACTIONS
  WHERE QUANTITY > 0
  GROUP BY SYMBOL
),
 SOLD(SYMBOL, AMOUNT) AS
(
  SELECT SYMBOL, -SUM(QUANTITY) FROM AWS STOCK_TRANSACTIONS
  WHERE QUANTITY < 0
  GROUP BY SYMBOL
)
SELECT B.SYMBOL, B.AMOUNT AS BOUGHT, S.AMOUNT AS SOLD
FROM BOUGHT B, SOLD S
WHERE B.SYMBOL = S.SYMBOL
ORDER BY B.AMOUNT DESC
FETCH FIRST 5 ROWS ONLY
```

	SYMBOL	BOUGHT	SOLD
0	KO	694562	409390
1	CSCO	692836	408547
2	PFE	683095	393959
3	INTC	639324	368778
4	VZ	592401	340565

# Connect with REST to Manage Data Virtualization

## Use advanced Microservices

```
In [5]: 1 # Connect to the Db2 Data Management Console service
2 Console  = 'https://zen-cpd-zen.apps.sre-dal10-demosundari-ga-02.demo.ibmcloud.com'
3 user     = 'admin'
4
5 # Set up the required connection
6 databaseAPI = Db2(Console)
7 api = '/v1'
8 databaseAPI.authenticate(api, user, password)
9 database = Console
```

```
In [6]: 1 r = databaseAPI.getDataSources()
2 print(r)
3 if (databaseAPI.getStatusCode(r)==200):
4     json = databaseAPI.getJSON(r)
5     df = pd.DataFrame(json_normalize(json))
6     print(', '.join(list(df)))
7     display(df)
8 else:
9     print(databaseAPI.getStatusCode(r))
```

```
<Response [200]>
agent_class, dataSources, dscount, hostname, is_docker, node_description, node_name, os_user, port
```

agent_class		dataSources	dscount	hostname	is_docker	node_description	node_name	os_user	port	
0	H		None	0	dv-0	N	Not specified	bigsq1	6414	
1	H	[{"cid": "DB210000", "dbname": "BANK", "srchost": "10.10.10.10", "srvt": "50000", "t": "H"}, {"cid": "DB210001", "dbname": "MORTGAGE", "srchost": "10.10.10.10", "srvt": "50001", "t": "H"}]	1	dv-0	N	Not specified	qpendpoint_3:6417	bigsq1	6417	
2	H		None	0	dv-0	N	Not specified	qpendpoint_4:6418	bigsq1	6418
3	H		None	0	dv-0	N	Not specified	qpendpoint_2:6416	bigsq1	6416
4	H		None	0	dv-0	N	Not specified	qpendpoint_5:6419	bigsq1	6419
5	H	[{"cid": "DB210003", "dbname": "MORTGAGE", "srchost": "10.10.10.10", "srvt": "50001", "t": "H"}]	1	dv-0	N	Not specified	qpendpoint_1:6415	bigsq1	6415	

# Connect applications to RESTful Endpoints

## Preview of the Db2 Endpoint Service

### Create a Unique RESTful Service

The most common way of interacting with the service is to fully encapsulate an SQL statement, including any parameters, in a unique RESTful service. This creates a secure separation between the database service and the RESTful programming service. It also allows you to create versions of the same service to make maintenance and evolution of programming models simple and predictable.

```
In [47]: API_makerest = "/v1/services"
```

Define the SQL that we want in the RESTful call.

```
In [51]: body = {"isQuery": True,
            "parameters": [
                {
                    "datatype": "VARCHAR(24)",
                    "name": "@CUSTID"
                }
            ],
            "schema": "STOCK",
            "serviceDescription": "Get customer payment",
            "serviceName": "getcustomerpay",
            "sqlStatement": "SELECT * FROM MONGODB.CUSTOMER_PAYMENT WHERE CUSTOMER_ID = @CUSTID",
            "version": "1.0"
        }
```

```
In [52]: try:
    response = requests.post("{}{}".format(Db2RESTful,API_makerest), headers=headers, json=body)
except Exception as e:
    print("Unable to call RESTful service. Error={}".format(repr(e)))
```

# Backup Slides



# Hybrid Cloud @ IBM Systems Center - Montpellier

January - Cloud Community

## Agenda

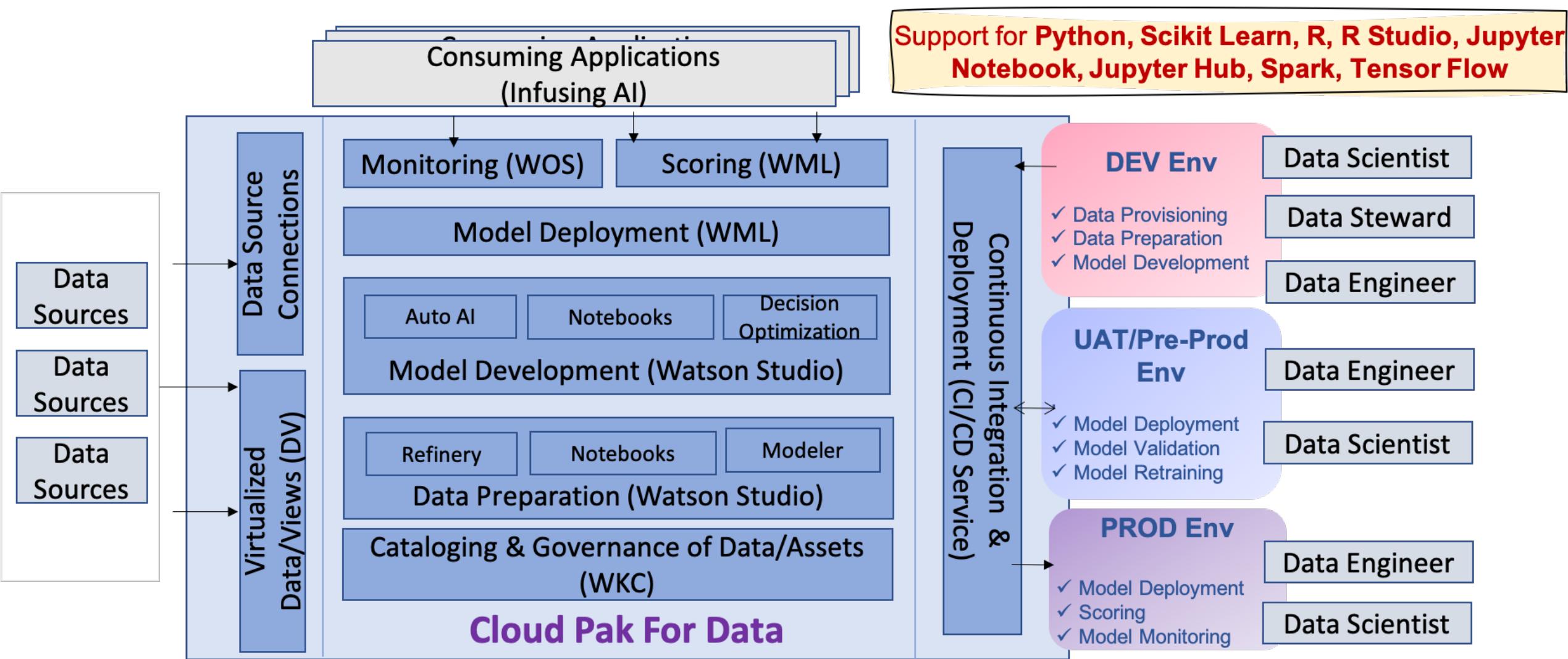
**10 AM - Friday, January 14<sup>th</sup>**  
<https://ibm.webex.com/meet/benoit.marolleau>

Support IBM Hybrid Cloud strategy with Systems capabilities



- ✓ Intro / Welcome
- ✓ MOP Network Hybrid Cloud status (AWS, IBM , ...)  
**Hervé Rey** - 5 minutes
- ✓ Available Infrastructure in 2022  
**Jerome Farenq** - 5 minutes
- ✓ CP4D/Data Fabric - Intro, Activities, Data virtualization  
**Benoit Marolleau** - 5 minutes
- ✓ Ansible Tower for Health Check & compliance  
**Charlotte Despres/Francis Cougard** - 15 minutes
- ✓ AIOps intro, solutions, contacts, Tech Zone Power Demo  
**Gauthier Siri** - 20 minutes
- ✓ Q&A / Call for Contributors / Interactions (cross dpt)  
Speakers, who's next?





 **Red Hat  
OpenShift**

 **Red Hat  
Enterprise Linux**

IBM public cloud



AWS



Microsoft Azure



Google Cloud



Private



IBM Z  
IBM LinuxOne  
IBM Power  
IBM Storage

Endpoints



**WS – Watson Studio**

**WML – Watson Machine Learning**

**WOS – Watson Open Scale**

**WKC – Watson Knowledge Catalog**

**DV – Data Virtualization**

**CI/CD – Continuous Integration & Deployment**

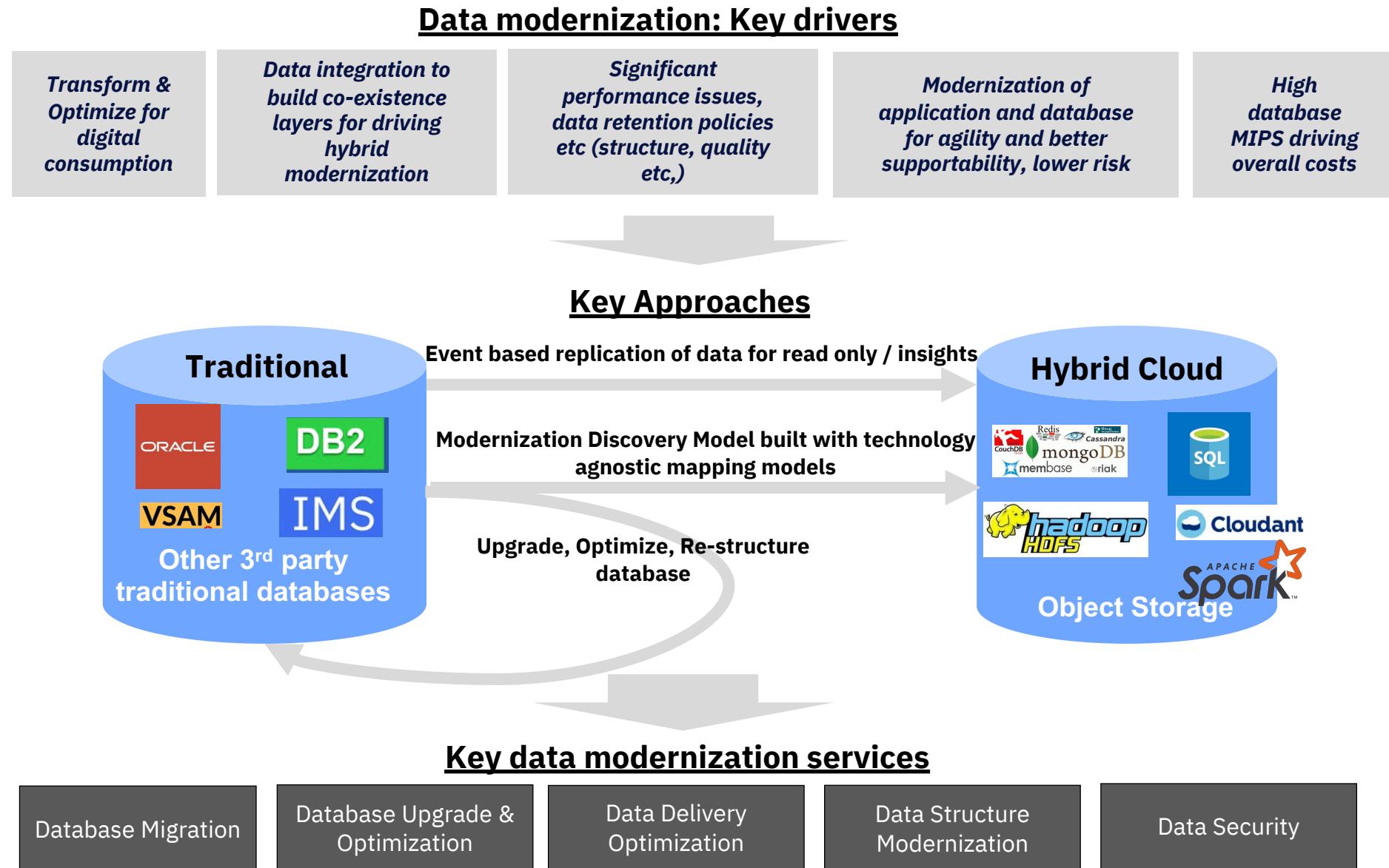
# Data Modernization and Transformation

Digital enablement, unlocking the data or cost optimization – always involved resolving data issues

Data issues have significant impact on scaling of any solution

A structured data modernization approach follows identifying the right data transformation / restructuring approach for the right use case

Deep understanding of the data is key to successful optimization of data



# Hybrid Data Modernization Patterns

Pattern	Description
Refactor	Re-architect and restructure existing data architecture leveraging a mix of new and existing data stores. Can include revised database type, DB schema, data model, etc. Simplest form of refactor would split up existing databases without changing the schema
Enrich	Combine new data sources with existing data to deliver advanced business value using BI, ML, and AI
Re-platform	Migrate database to new target. Make minimal changes to schema to adapt to the new platform, but do not change the structure or the features or functions it provides
Containerize	Move a database into a container-based version of the same database technology (with a minimal amount of application change)
Externalize	Provide API access to existing data, while preserving current single-source of truth
Migrate	Lift and shift database to a cloud environment without changing underlying DB technology (typically only used as an interim solution for modernization candidates), including changing encryption scheme
Replicate	Copy data to another availability zone for backup, to different regions for disaster recovery, to cache to optimize application access
Retain	Do nothing (could be left as is for one-to-many modernization increments or as a final disposition)