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
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 …

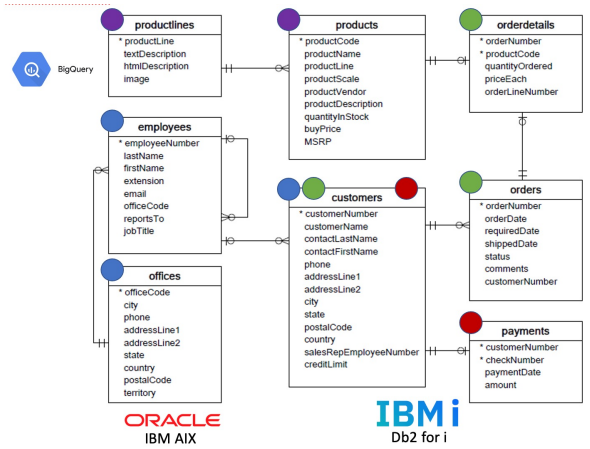
ETL Simplification with Data consumption - Apache NIFI (ETL) or Salesforce

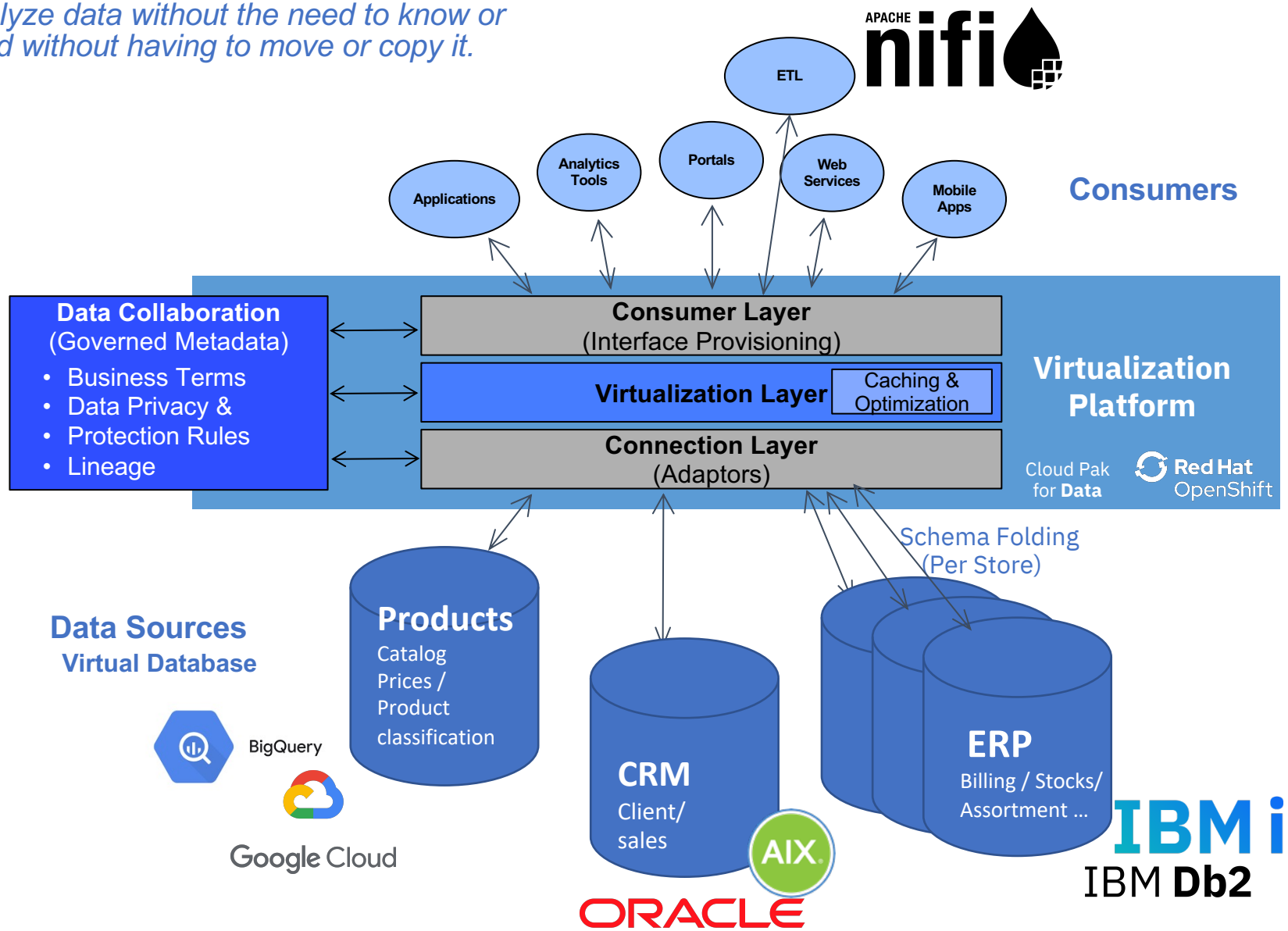
RetailOne Demo
IBM Data Fabric pattern: Governed Data Virtualization

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.

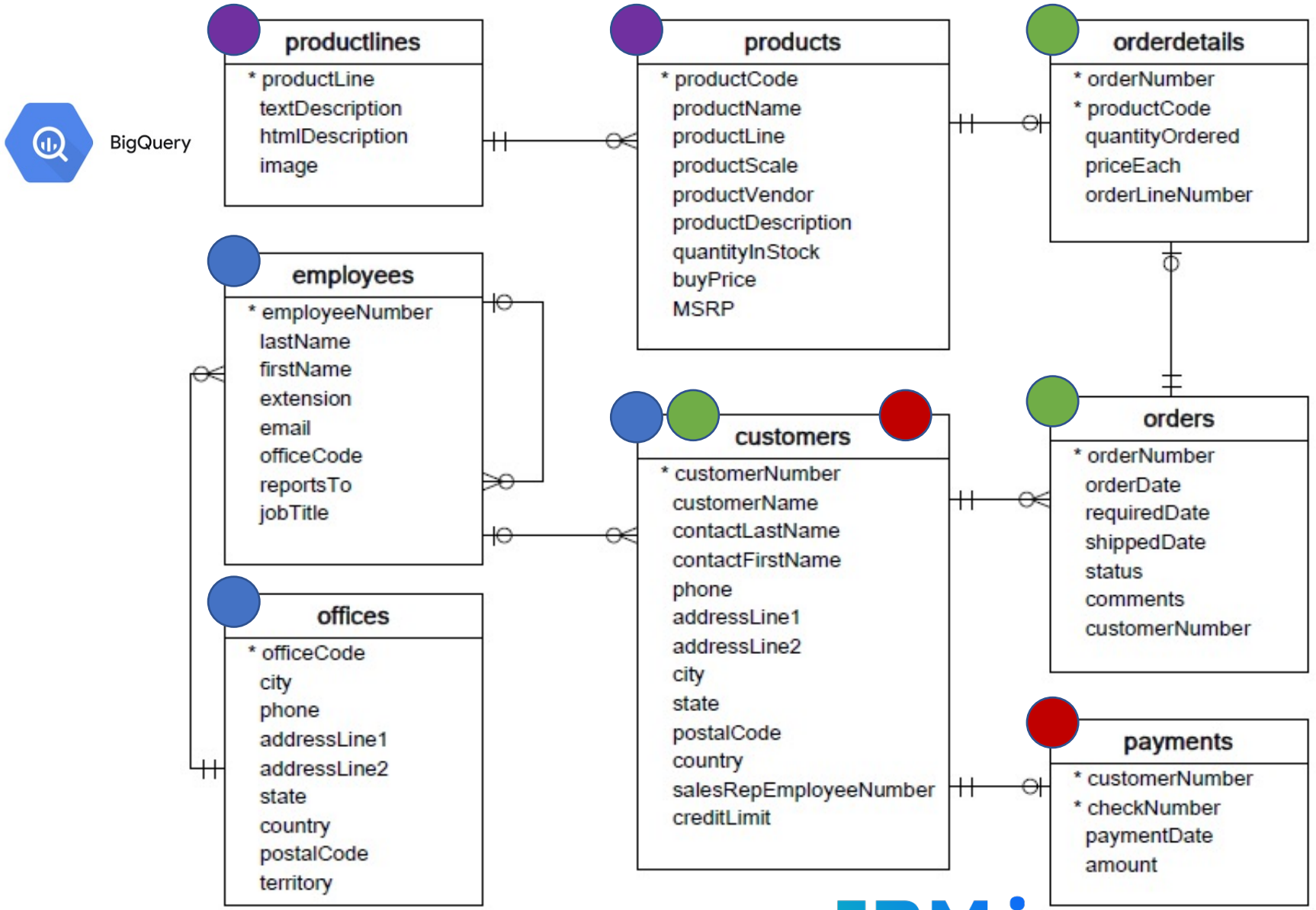
Impact of modern Information Architecture

- 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



ORACLE
IBM AIX

IBM i
Db2 for i

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

RetailOne Demo
IBM Data Fabric pattern: Governed Data Virtualization

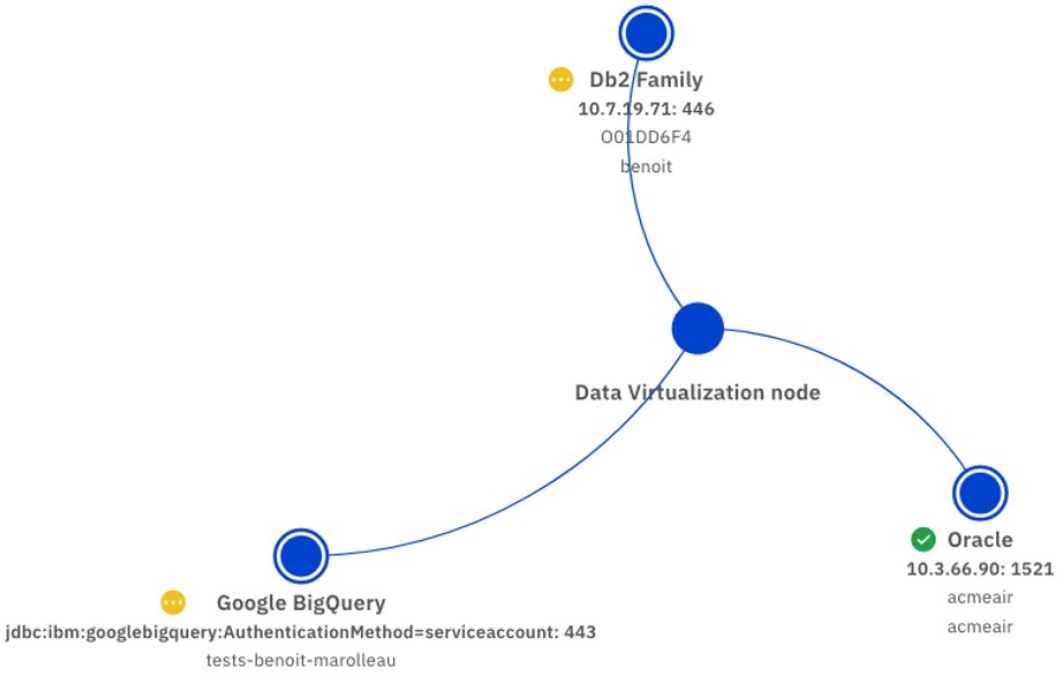
Data consist of the following databases/schemas/tables:

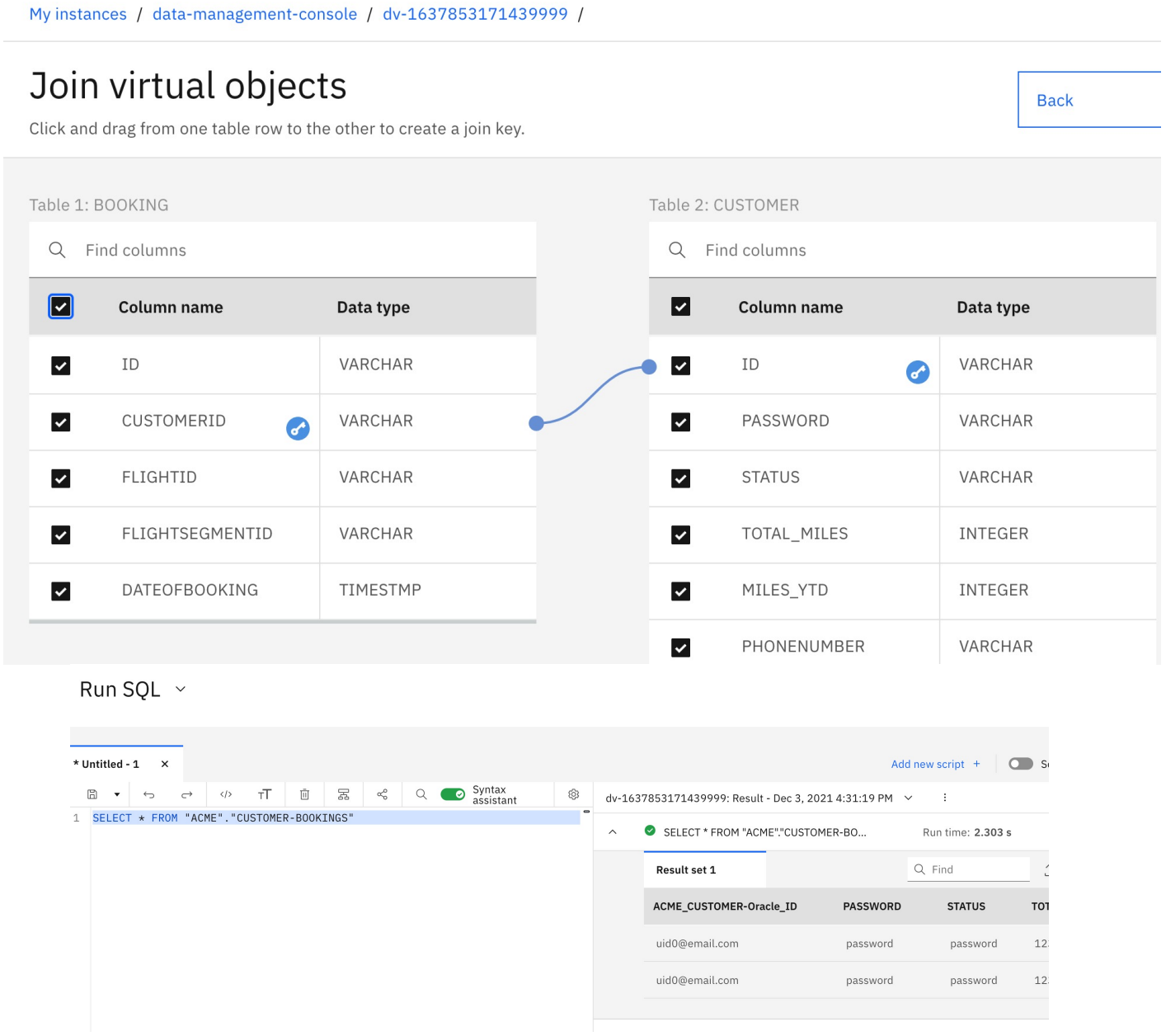
- Big Query - GCP
  - Products: stores a list of scale model cars.
  - ProductLines: stores a list of product line categories.

- 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.

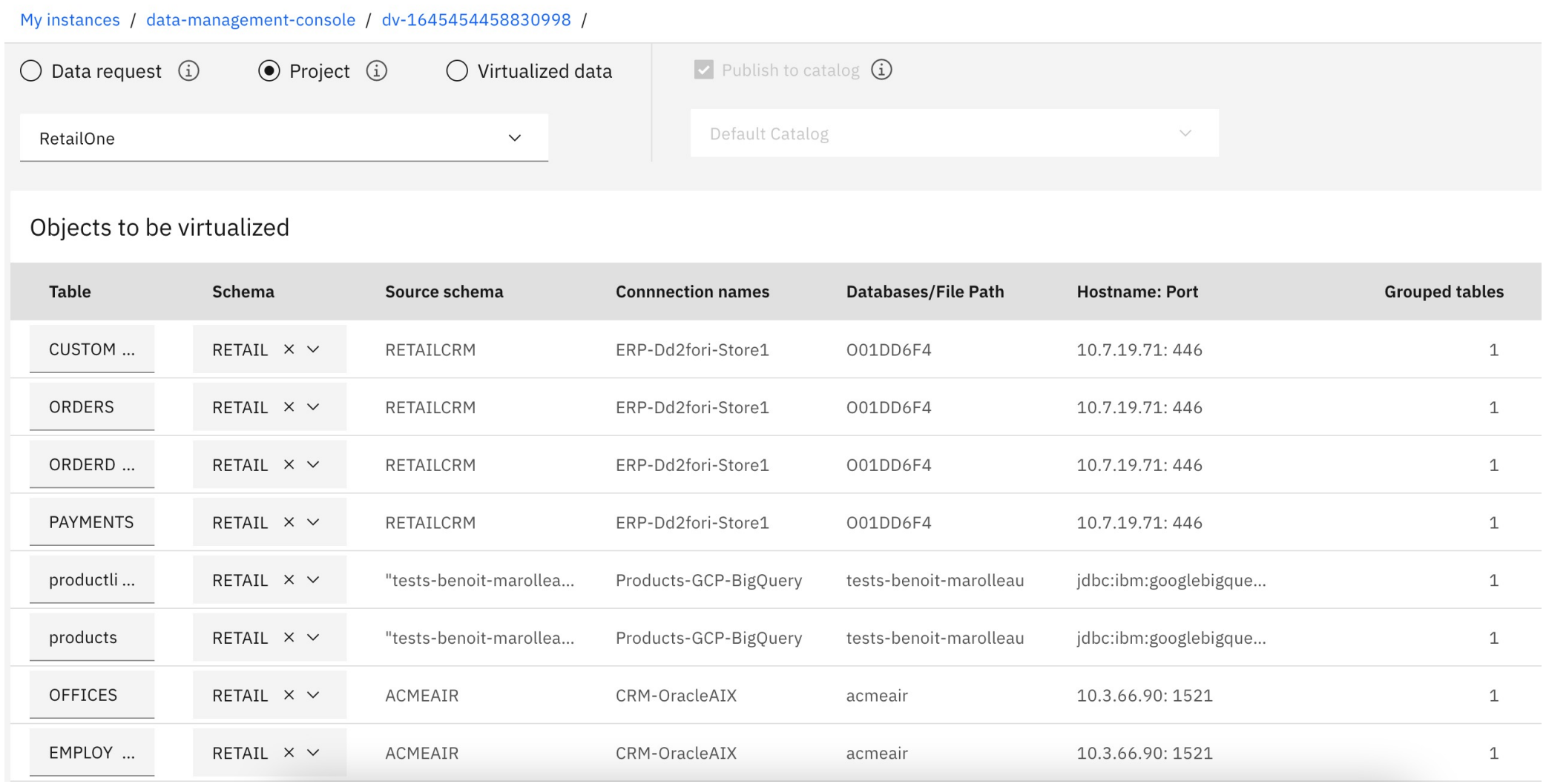
- Oracle AIX - On prems :
  - Employees: stores all employee information as well as the organization structure such as who reports to whom.
  - Offices: stores sales office data.

RetailOne Demo





RetailOne Demo



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)
- Reenforced 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

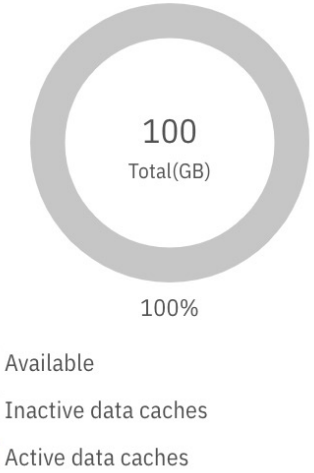
Data caches | Queries

Cache storage

Available
100.0 GB

Inactive data caches
0.0 KB

Active data caches
0.0 KB



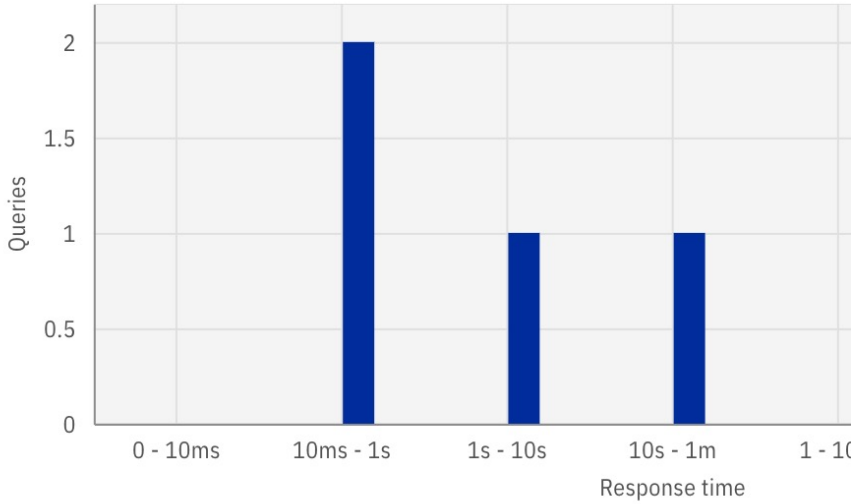
Responsiveness

Total queries
4

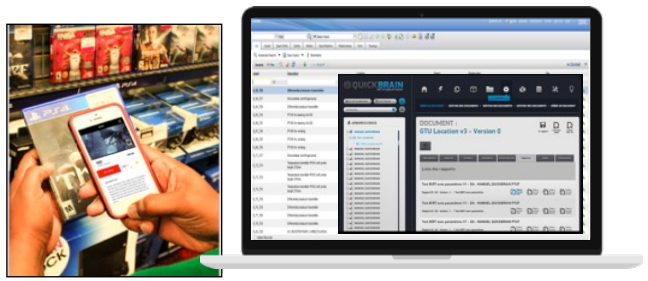
Avg response time
3s 880ms

Using cache
0.0%

Not using cache
100.0%



Hybrid Cloud Solutions



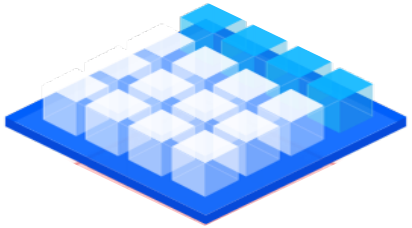
Retail / Banking use case

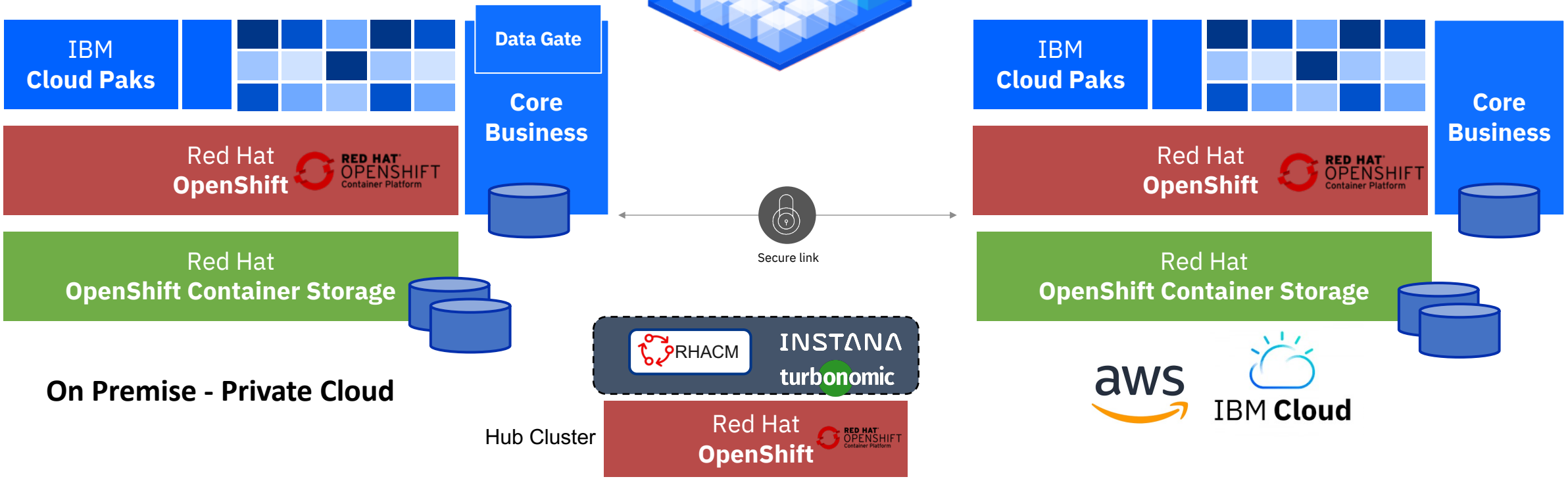
- Data Platform Modernization
- Data Compliance
- Customer Data Management

- AIOps
- Multi-Cloud Management
- Business Continuity

ANY DATA, ANY CLOUD, ANYWHERE

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





On Premise - Private Cloud

Cluster Lifecycle – App Lifecycle – Observability – Governance/Risk

Extend and Standardize your Data Estate

Connect heterogeneous data sources

Join data using advanced SQL

Create views to simplify access

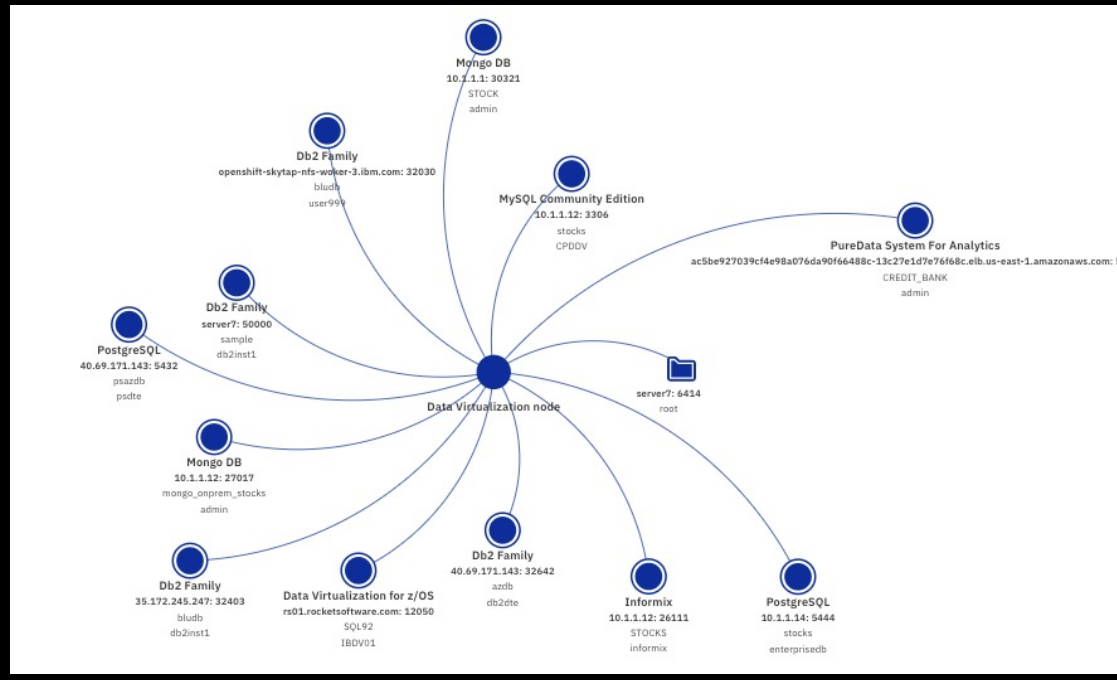
Protect data with deep data governance

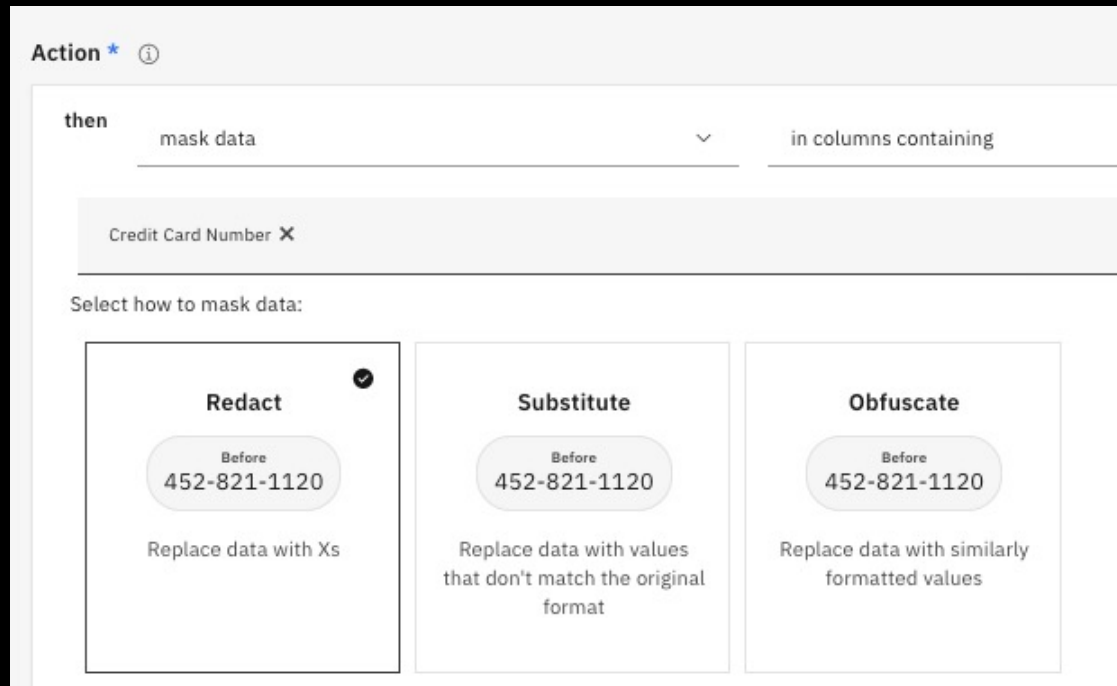
Accelerate performance with local caching

Connect with your existing tools

Build like a Data Scientist with Watson Studio

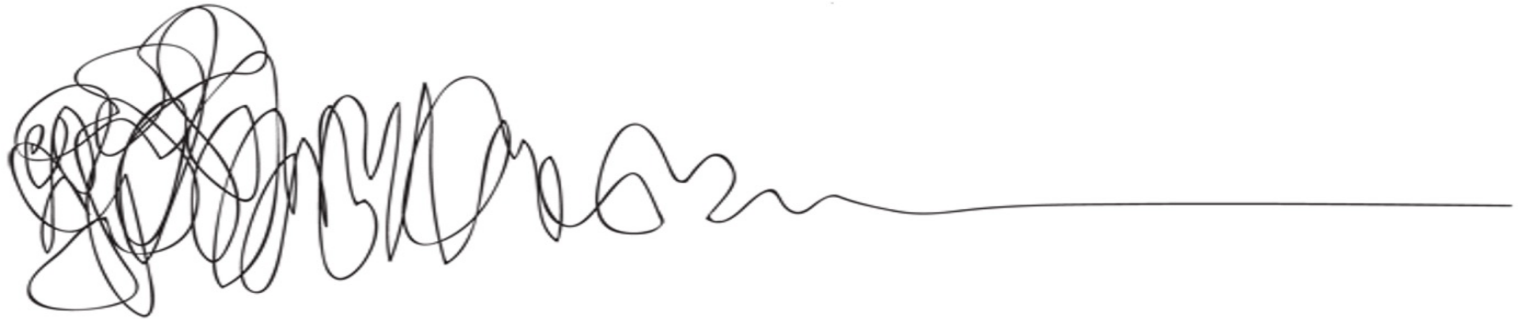
Access through RESTful services and endpoints



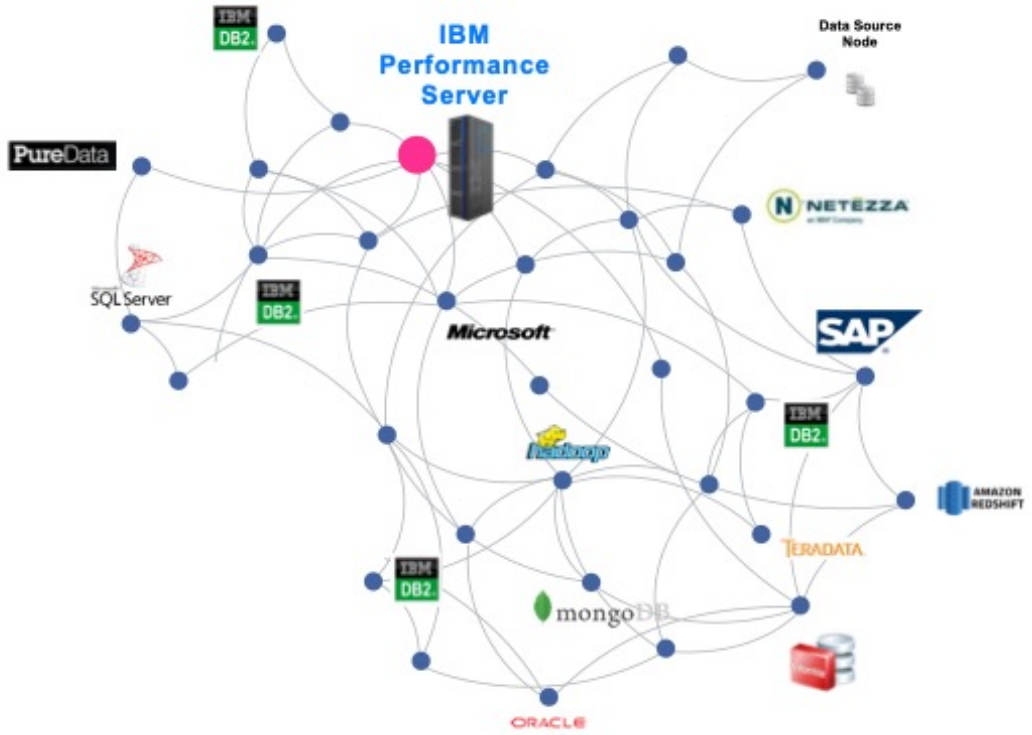


Why is it **harder than it looks?**

- 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**

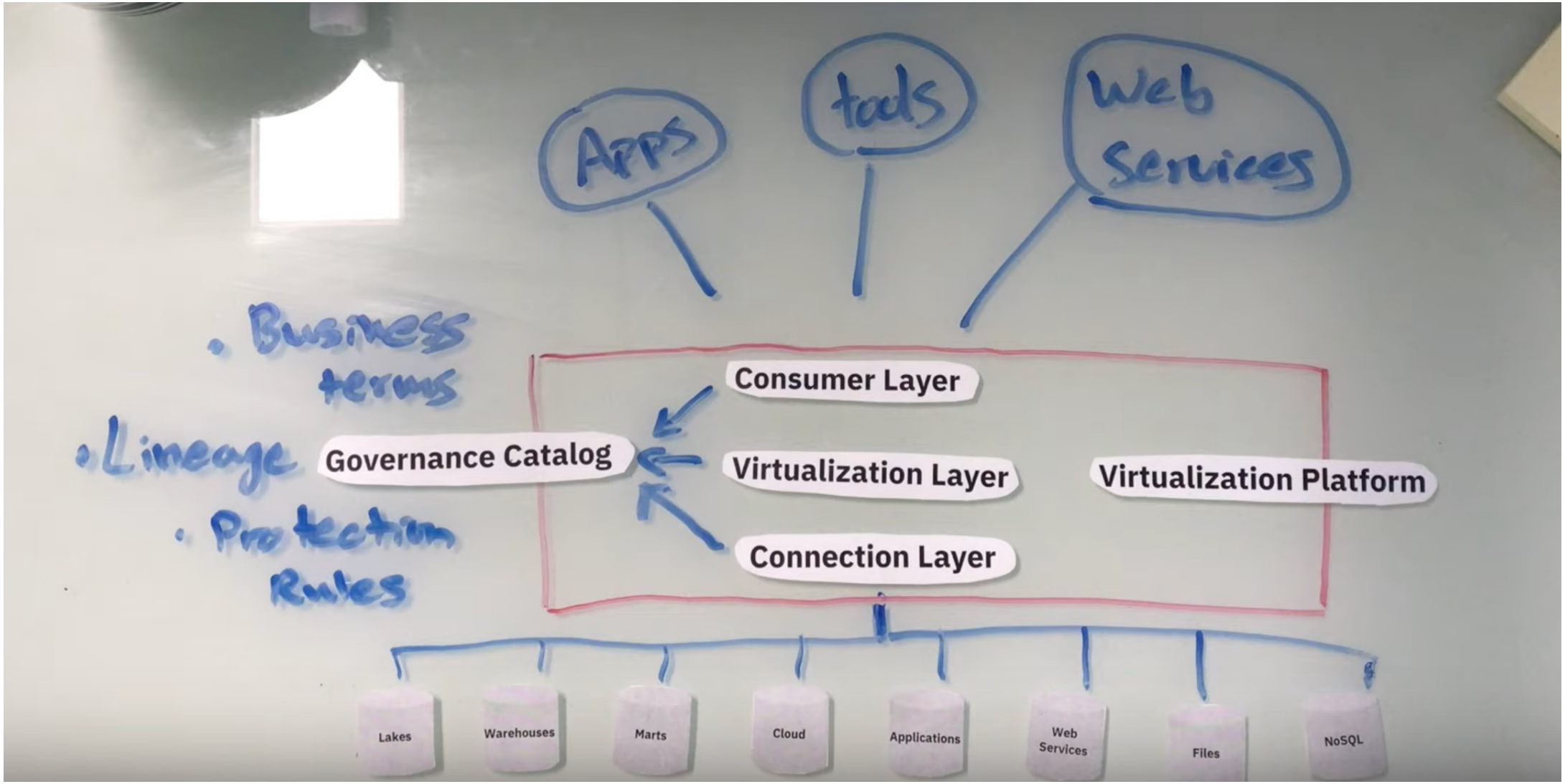


Look like **this**

With **this**

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

Data Virtualization



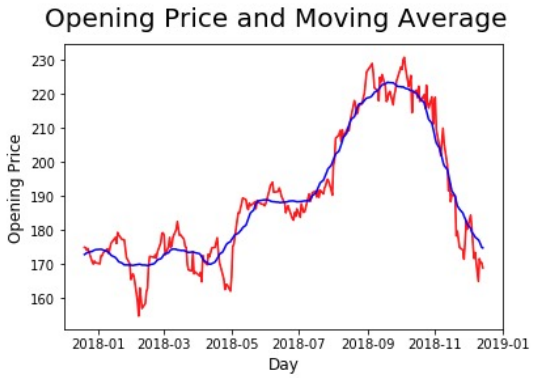
All your Data Analysis from one Data Source

One connection for simplicity and performance

30 Day Moving Average of Apple Stock

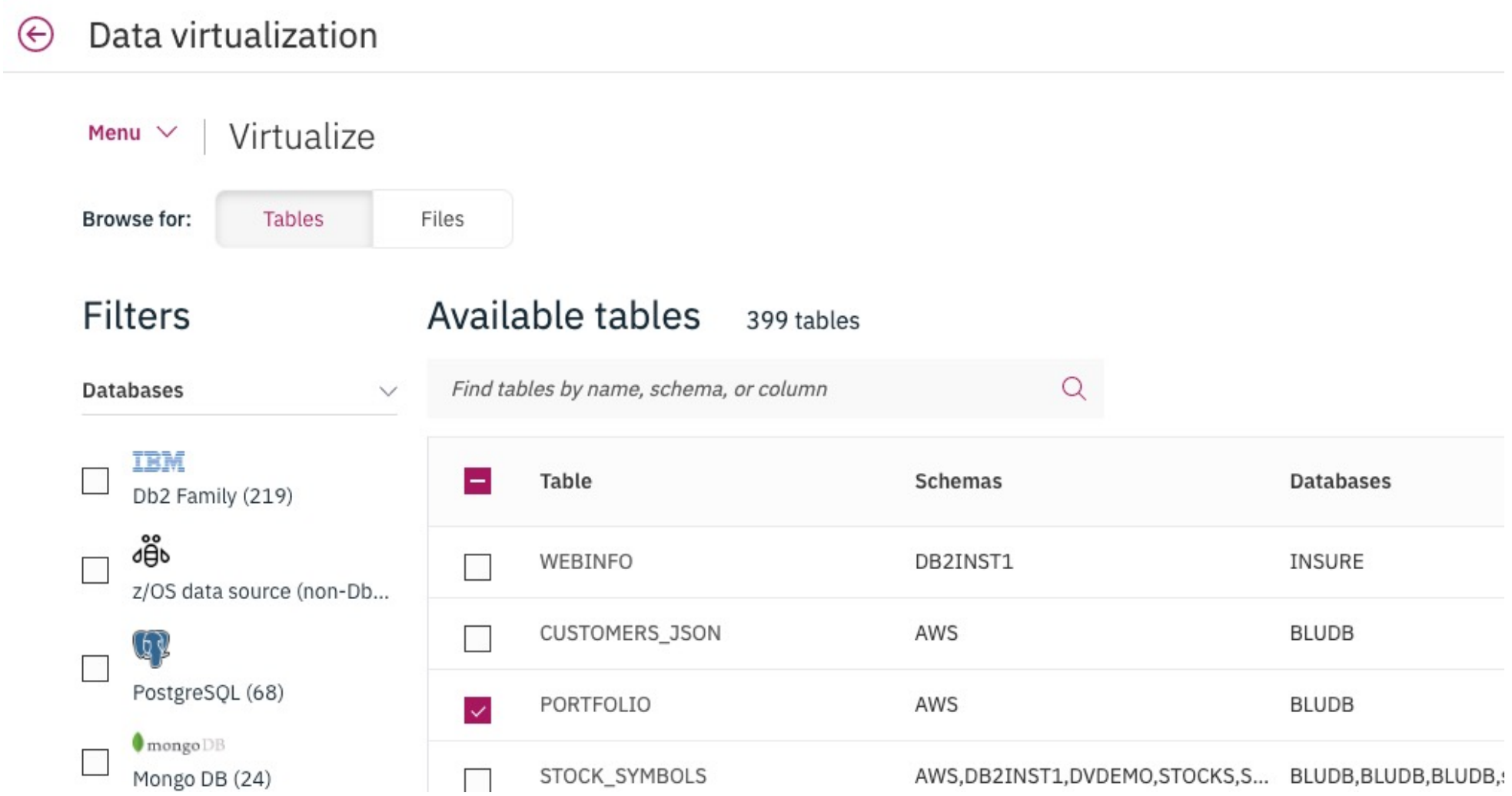
In [7]:
```
sqlin = \
"""
SELECT TX_DATE, OPENING,
     AVG(OPENING) OVER (
       ORDER BY TX_DATE
     ROWS BETWEEN 15 PRECEDING AND 15 FOLLOWING) AS MOVING_AVG
  FROM AWS.STOCK_HISTORY
     WHERE SYMBOL = 'AAPL'
  ORDER BY TX_DATE
"""
df = %sql {sqlin}
txdate= df['TX_DATE']
sales = df['OPENING']
avg = 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();
```



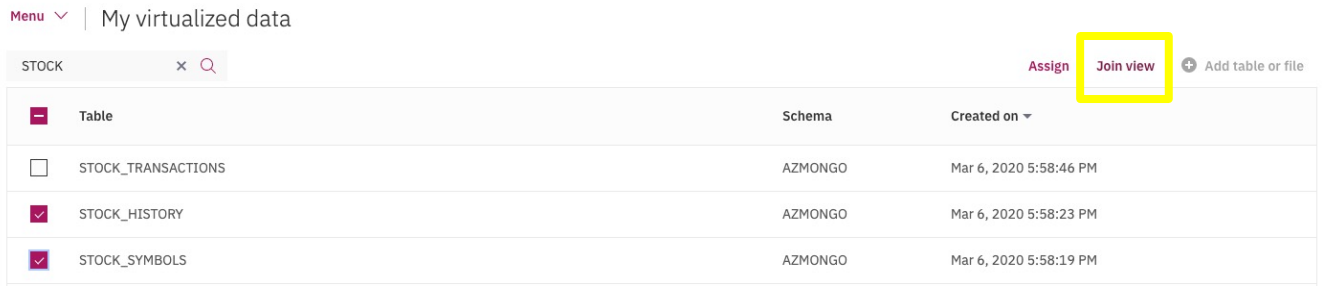
Virtualize Data

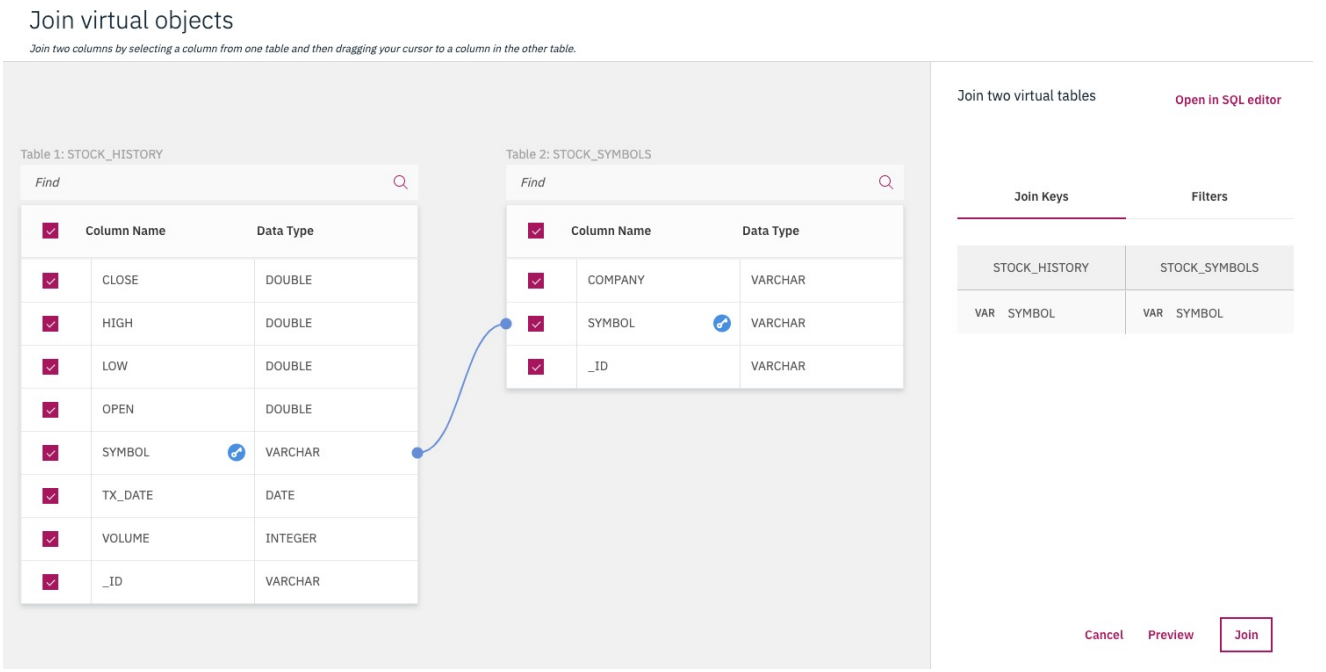
Make a remote table work like a local table without moving data



Join virtual tables

Make two virtual tables work like a single local table

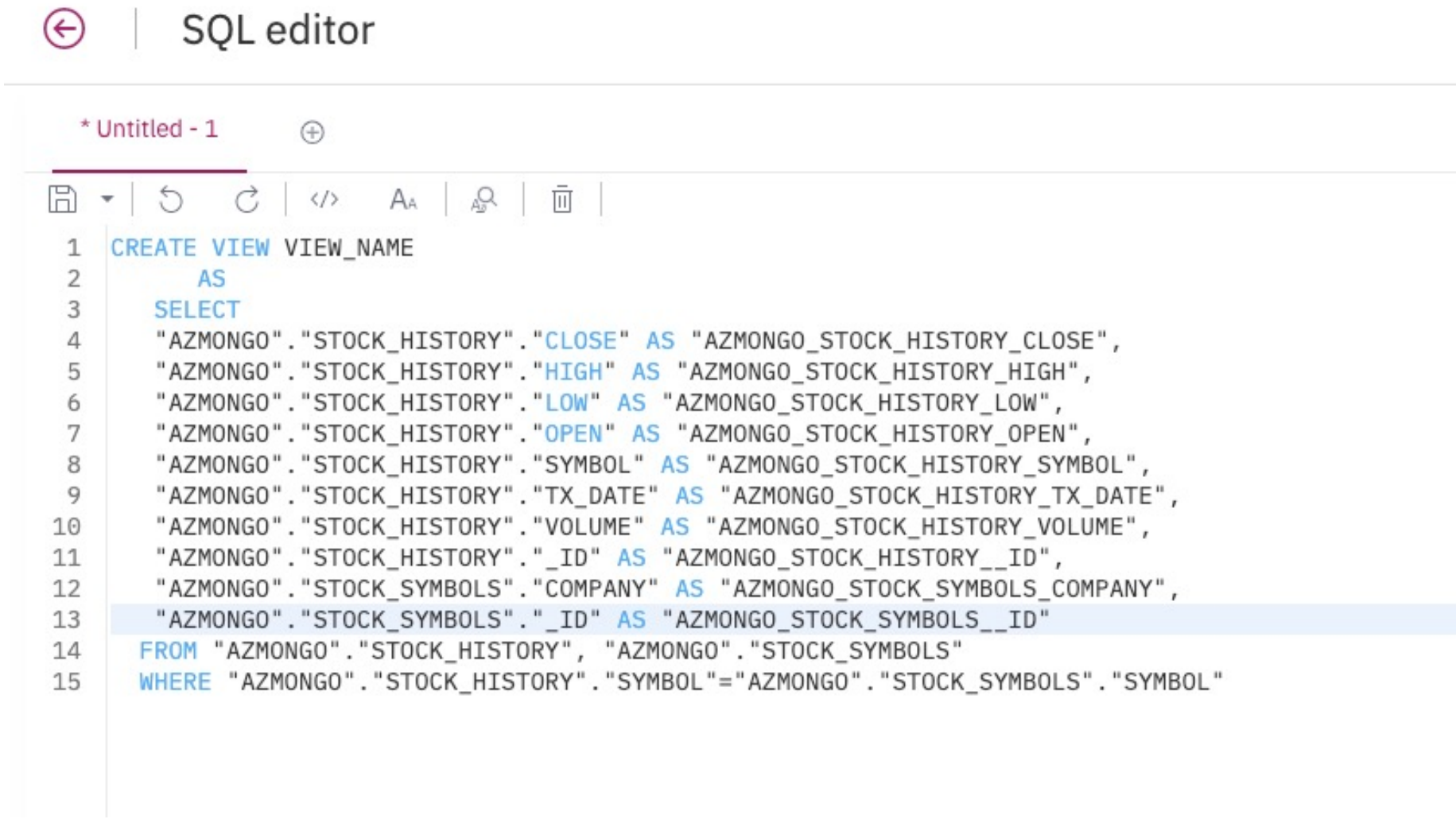




SQL editor

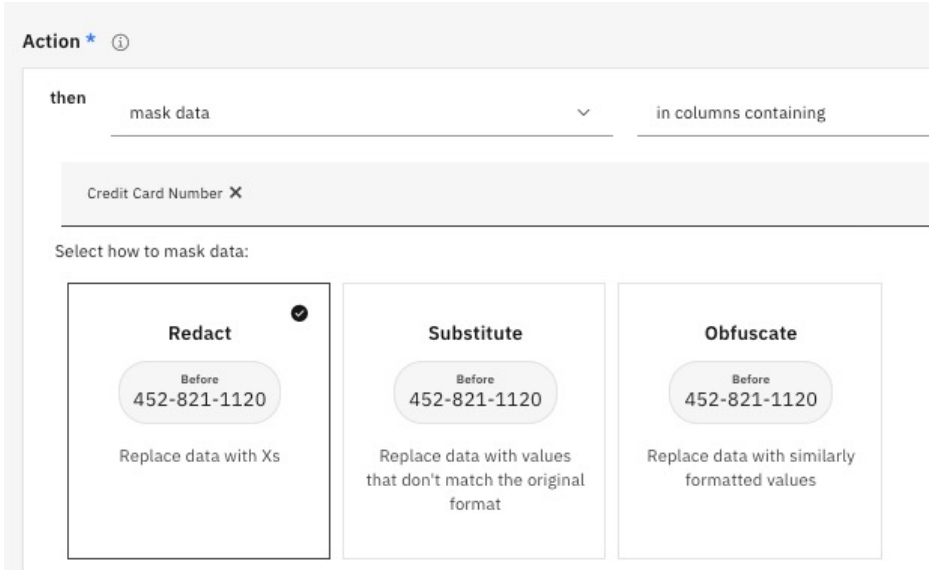
Build Views from Complex Queries

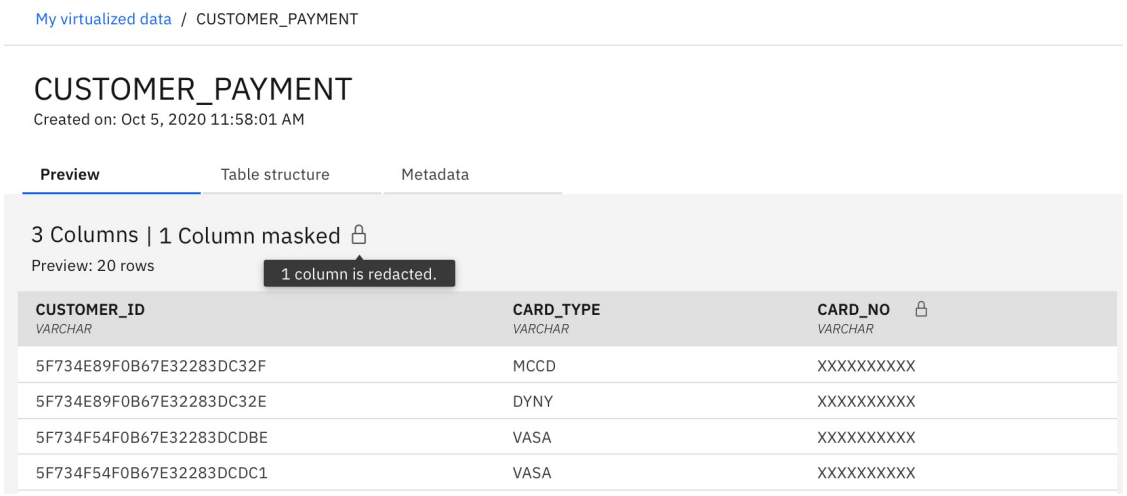
Design your own queries so they work like a local table



Protect Data using Deep Data Governance

Classify data and protect using active masking



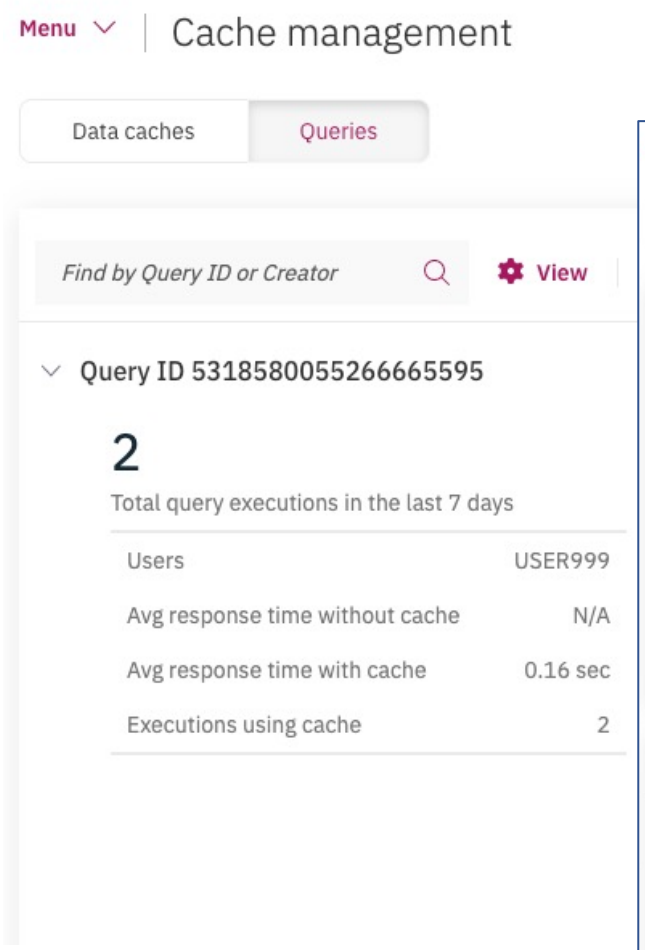


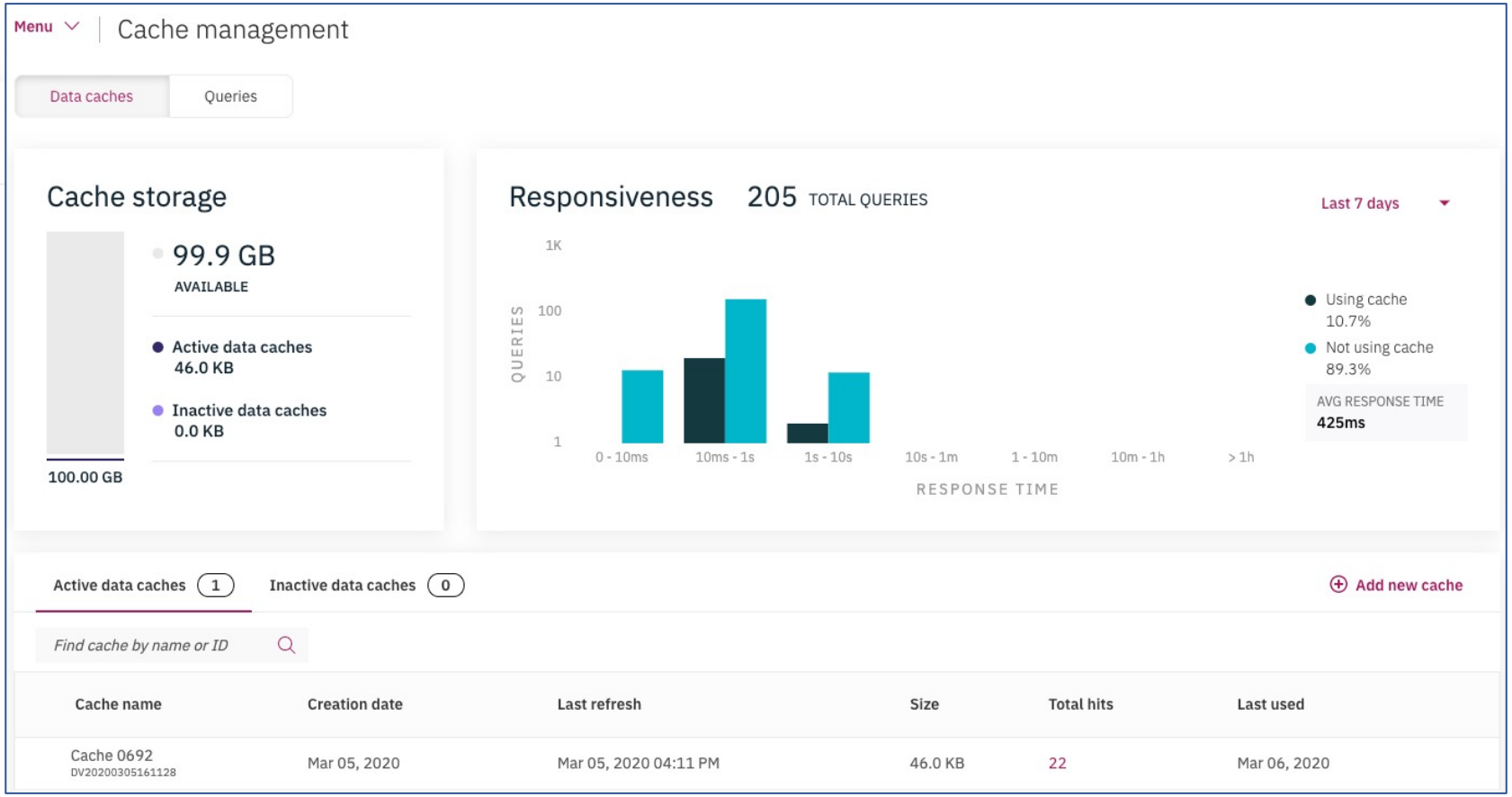

```
In [4]: %sql SELECT * from STOCK.CUSTOMER_PAYMENT FETCH FIRST 5 ROWS ONLY
Out[4]:
```

	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

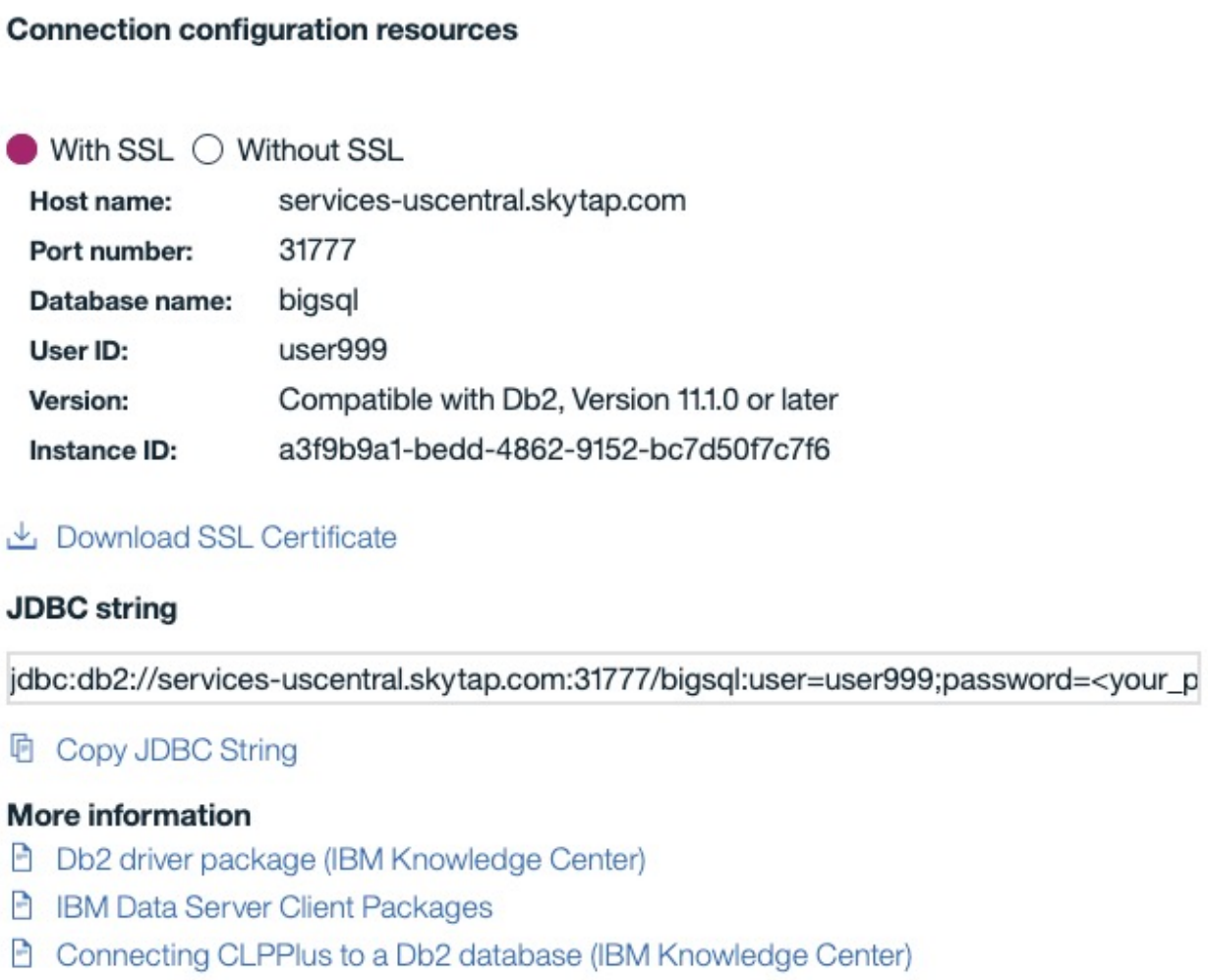
Optimize

Cache results for optimum performance





Connect your Analytics Applications



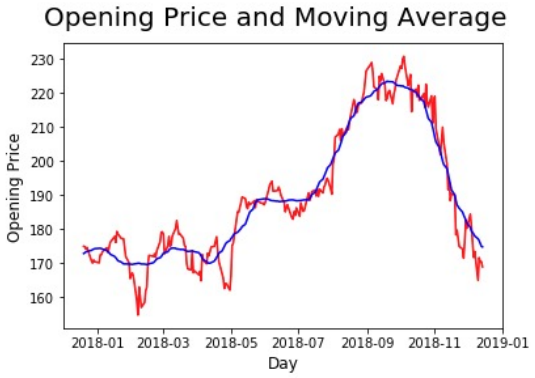
Connect with Jupyter Hub

Use advanced Data Scientist Tools

30 Day Moving Average of Apple Stock

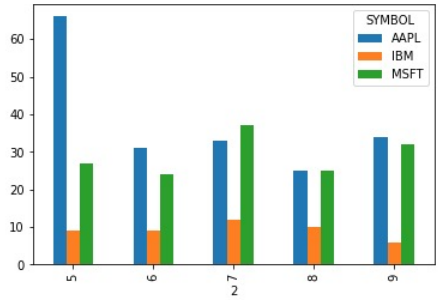
In [7]:
```
sqlin = \
"""
SELECT TX_DATE, OPENING,
     AVG(OPENING) OVER (
       ORDER BY TX_DATE
     ROWS BETWEEN 15 PRECEDING AND 15 FOLLOWING) AS MOVING_AVG
  FROM AWS.STOCK_HISTORY
     WHERE SYMBOL = 'AAPL'
  ORDER BY TX_DATE
"""
df = %sql {sqlin}
txdate= df['TX_DATE']
sales = df['OPENING']
avg = 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();
```



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
```

Out[9]:

	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]:

```
# Connect to the Db2 Data Management Console service
Console  = 'https://zen-cpd-zen.apps.sre-dal10-demosundari-ga-02.demo.ibmcloud.com'
user     = 'admin'

# Set up the required connection
databaseAPI = Db2(Console)
api = '/v1'
databaseAPI.authenticate(api, user, password)
database = Console
```

In [6]:

```
r = databaseAPI.getDataSources()
print(r)
if (databaseAPI.getStatusCode(r)==200):
    json = databaseAPI.getJSON(r)
    df = pd.DataFrame(json_normalize(json))
    print(', '.join(list(df)))
    display(df)
else:
    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	AdminNode	bigsql	6414
1	H	[{'cid': 'DB210000', 'dbname': 'BANK', 'srchos...	1	dv-0	N	Not specified	qpendpoint_3:6417	bigsql	6417
2	H	None	0	dv-0	N	Not specified	qpendpoint_4:6418	bigsql	6418
3	H	None	0	dv-0	N	Not specified	qpendpoint_2:6416	bigsql	6416
4	H	None	0	dv-0	N	Not specified	qpendpoint_5:6419	bigsql	6419
5	H	[{'cid': 'DB210003', 'dbname': 'MORTGAGE', 'sr...	1	dv-0	N	Not specified	qpendpoint_1:6415	bigsql	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

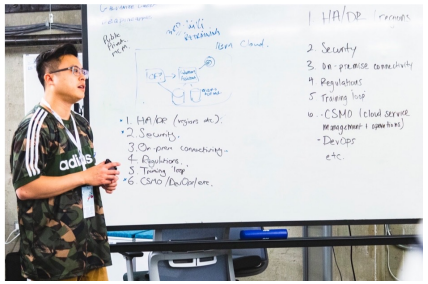
January - Cloud Community

Agenda

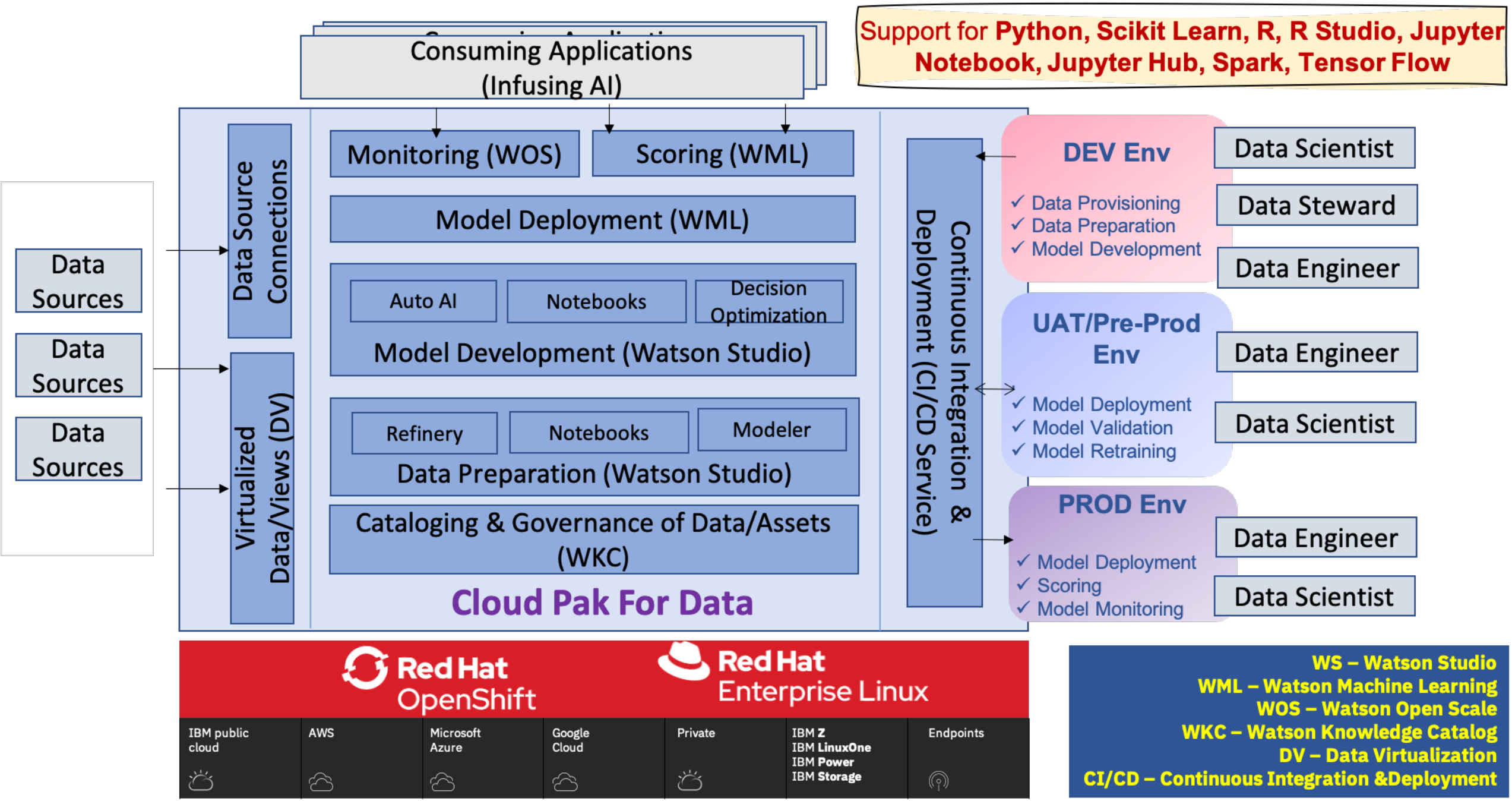
10 AM - Friday, January 14th
https://ibm.webex.com/meet/benoit.marolleau

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?



ANSIBLE


Support for Python, Scikit Learn, R, R Studio, Jupyter Notebook, Jupyter Hub, Spark, Tensor Flow
Consuming Applications (Infusing AI)
Data Sources
Data Sources
Data Sources
Data Source Connections
Virtualized Data/Views (DV)
Monitoring (WOS)
Scoring (WML)
Model Deployment (WML)
Auto AI
Notebooks
Decision Optimization
Model Development (Watson Studio)
Refinery
Notebooks
Modeler
Data Preparation (Watson Studio)
Cataloging & Governance of Data/Assets (WKC)
Cloud Pak For Data
Continuous Integration & Deployment (CI/CD Service)
DEV Env
✓ Data Provisioning
✓ Data Preparation
✓ Model Development
Data Scientist
Data Steward
Data Engineer
UAT/Pre-Prod Env
✓ Model Deployment
✓ Model Validation
✓ Model Retraining
Data Engineer
Data Scientist
PROD Env
✓ Model Deployment
✓ Scoring
✓ Model Monitoring
Data Engineer
Data Scientist
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

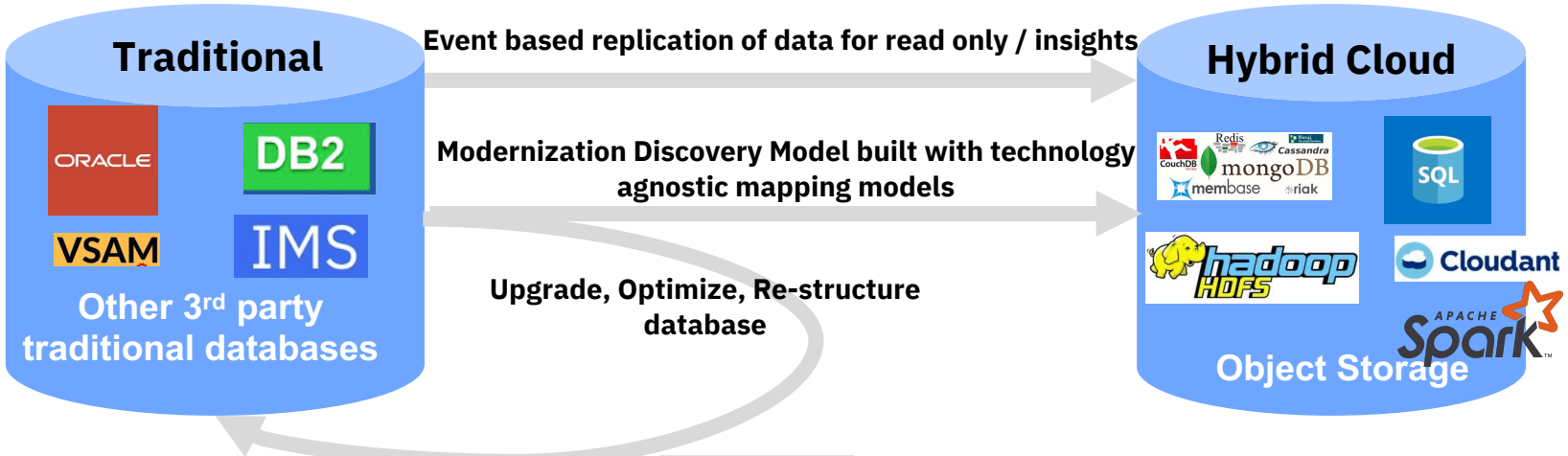
A structured data modernization approach follows identifying the right data transformation / restructuring approach for the right use case

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

Data modernization: Key drivers

Transform & Optimize for digital consumption	*Data integration to build co-existence layers for driving hybrid modernization*	*Significant performance issues, data retention policies etc (structure, quality etc,)*	*Modernization of application and database for agility and better supportability, lower risk*	*High database MIPS driving overall costs*

Key Approaches



Key data modernization services

Database Migration	Database Upgrade & Optimization	Data Delivery Optimization	Data Structure Modernization	Data Security

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)