Insight | Digital Innovation

Databricks, an Introduction

Chuck Connell, Insight Digital Innovation

Insight Presentation

Speaker Bio

- Senior Data Architect at Insight Digital Innovation
- Focus on Azure big data services – HDInsight/Hadoop, Databricks, Cosmos DB
- Related work…
  - NoSQL and relational data models, transitions
  - Size/volume estimates
  - JSON schemas for schema-less data
- Boston office in Watertown Sq, on the river walk

Creating meaningful connections that help businesses run smarter.

Supply Chain Optimization

We help you invest smarter so you can manage today and transform the future.

Connected Workforce

We create a connected workplace so employees can work smarter.

Cloud & Data Center Transformation

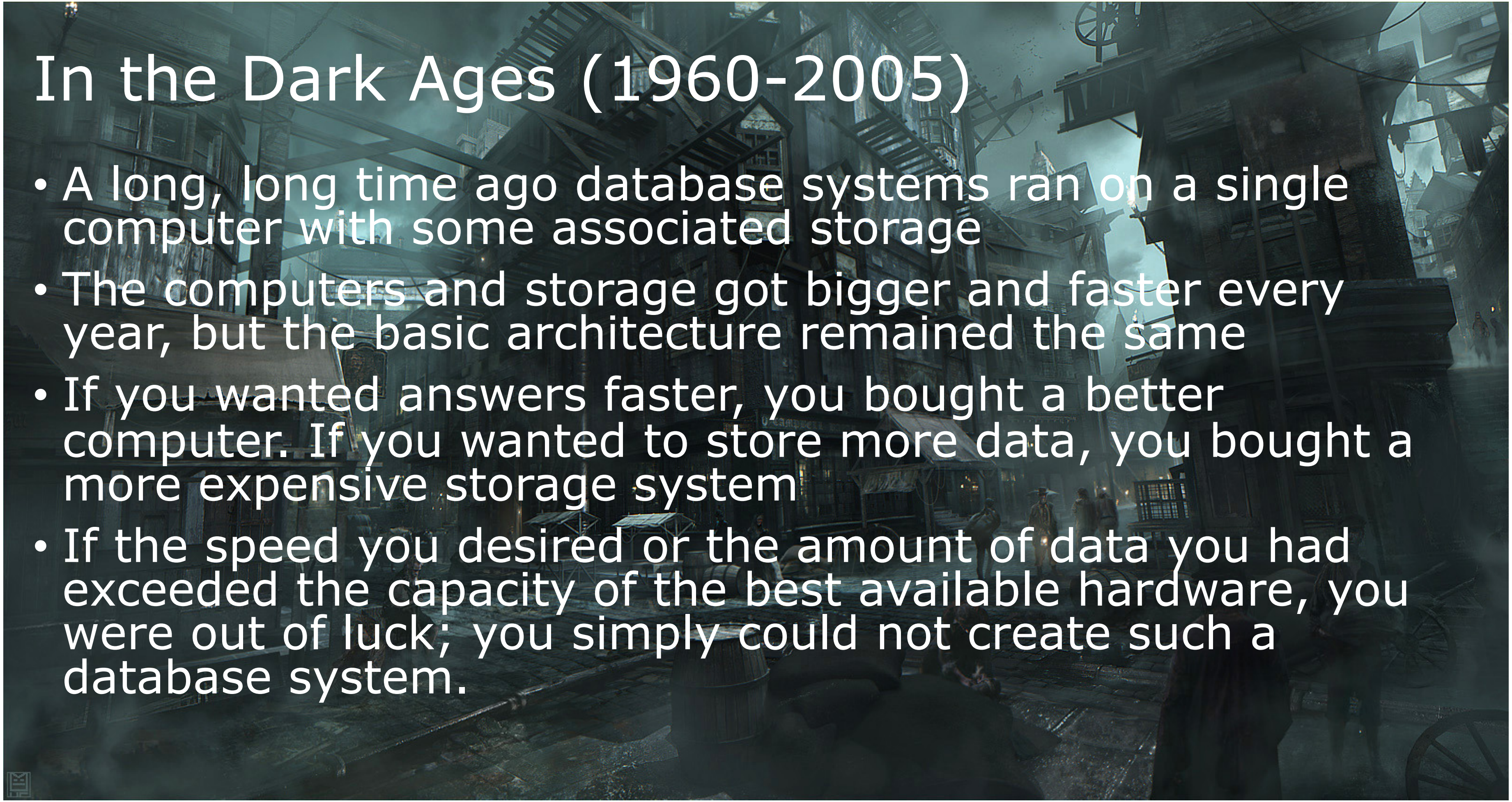
We help you prepare for the future and align workloads to the right platforms.

Digital Innovation

We help you innovate smarter so you can make meaningful connections.

Talk Outline

- Brief history -- what came before, why Databricks
- Spin up a Databricks instance, verify it
- Add some data
- SQL table operations
- DataFrame operations
- DB “connections”, getting data in and out
- Other cool things you can do with Databricks
- Caveats – what is not perfect about Databricks
- Q&A



In the Dark Ages (1960-2005)

- A long, long time ago database systems ran on a single computer with some associated storage
- The computers and storage got bigger and faster every year, but the basic architecture remained the same
- If you wanted answers faster, you bought a better computer. If you wanted to store more data, you bought a more expensive storage system
- If the speed you desired or the amount of data you had exceeded the capacity of the best available hardware, you were out of luck; you simply could not create such a database system.

Hadoop

- In 2006, Doug Cutting et al at Yahoo created Hadoop
  - Unlimited horizontal scaling on cheap computers!
- Key ideas…
  - HDFS, all disks became one file system
  - MapReduce, a way to run parallel code on all the CPUs
- Soon there were Hadoop clusters with 100s of nodes, then 1000s
- You could do database things that were simply impossible before!
- But there is no free lunch
  - What were the main drawbacks to Hadoop?

Hadoop Drawbacks

- MapReduce is hard to program
  - A new way of thinking about coding
  - Does not magically parallelize algorithms for you
  - Requires trial/error, tuning, multiple stages of MR
- Hadoop wrote MR intermediate results to disk
  - Often many times for one job
  - Much slower than a memory write/read
- Hadoop was a “batch” system, not interactive queries

Hive

- Introduced in 2009, it solved the "hard to program" problem
- SQL abstraction on top of HDFS and MapReduce
  - Data appears as normal relational-like tables
  - Database jobs can be written in SQL
- Essentially a compiler that translates SQL into Java MapReduce code
  - Generated code usually better than human would create
- But still… lots of disk I/O
  - Simple Hive query on small table = ~15 secs

Spark

- In 2011, Spark project to solve Hadoop disk I/O problem
- Goal: do as many operations as possible completely within memory
- Spark delivered 10 – 100x speedup on fewer machines
- But alas, still no free lunch
    - What are the key problems with Spark?

Spark, Issues

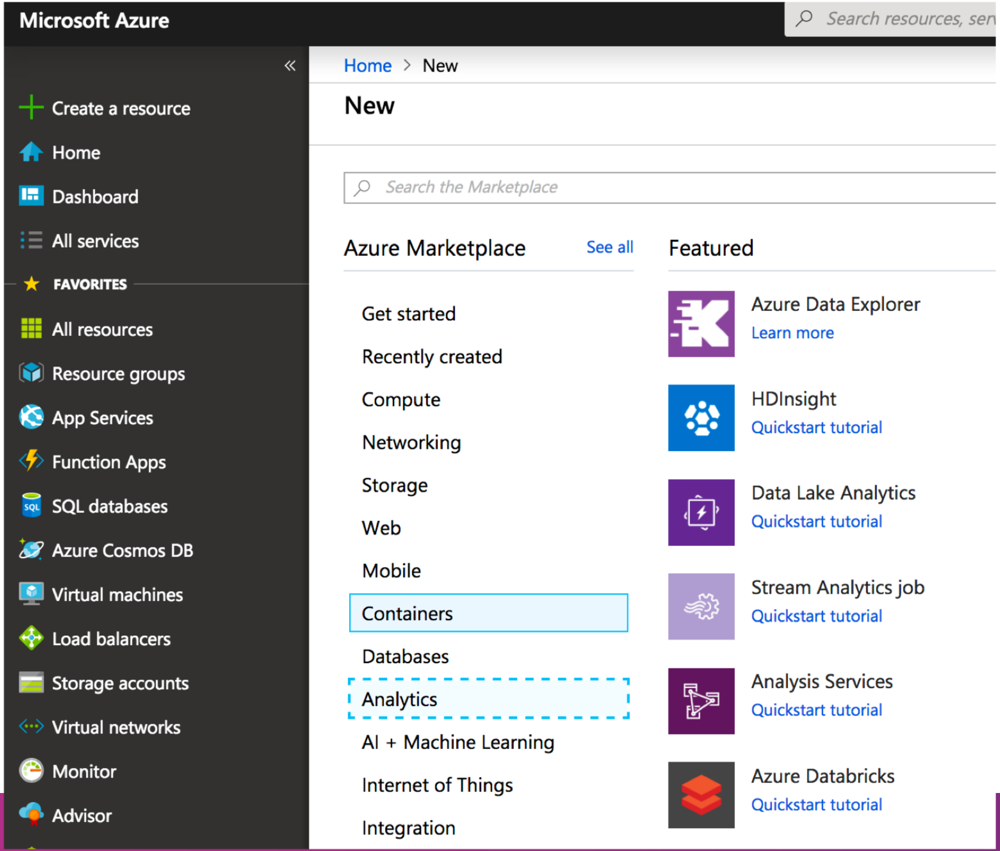
- Complexity
  - Software installs
  - Hardware clusters
  - File system setup
  - Performance tuning
- Security
  - Clusters
  - Code / Jobs
  - Data
- Enter Databricks (2015)

Databricks

- Databricks is a way to use Spark more conveniently
- Databricks *is* Spark, but with a GUI and many automated features
  - Creation and configuration of server clusters
  - Auto-scaling and shutdown of clusters
  - Connections to various file systems and formats
  - Programming interfaces for Python, Scala, SQL, R
  - Integration with other Azure services
- Available only as a cloud service, both Amazon and Azure

- Let’s dive in…

Start Databricks, Azure



Microsoft Azure

Search resources, serv

«

- Create a resource
- Home
- Dashboard
- All services
- ★ FAVORITES
- All resources
- Resource groups
- App Services
- Function Apps
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Monitor
- Advisor

Home > New

New

Search the Marketplace

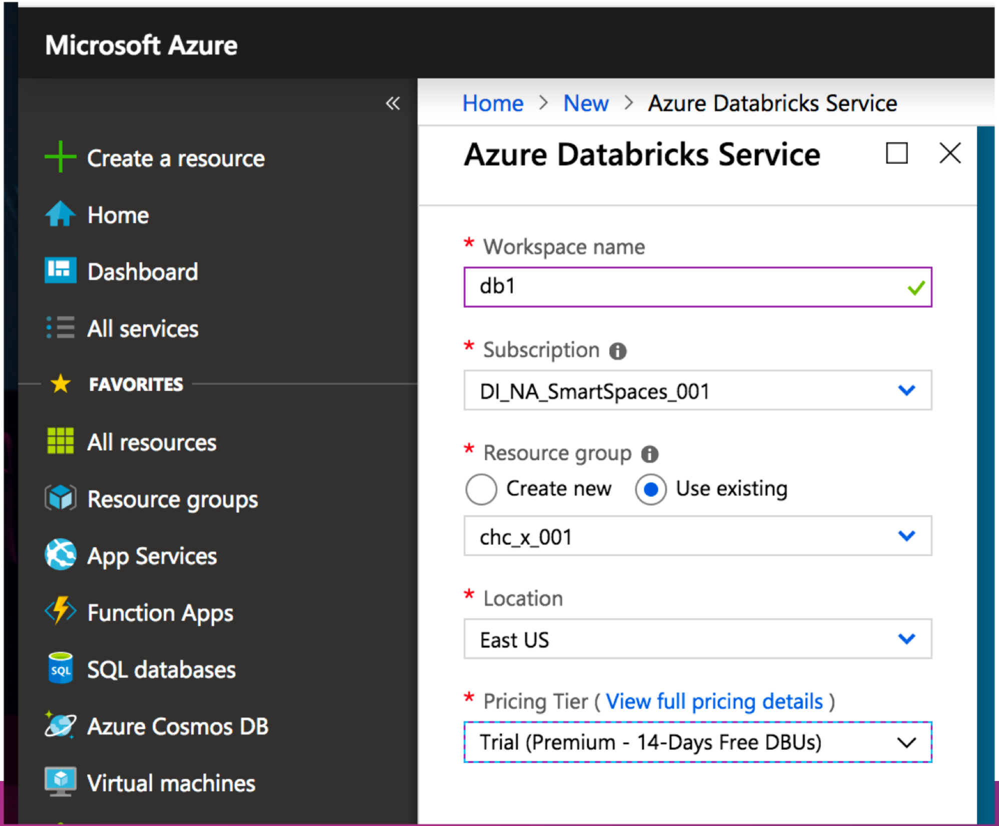
Azure Marketplace See all

- Get started
- Recently created
- Compute
- Networking
- Storage
- Web
- Mobile
- Containers
- Databases
- Analytics
- AI + Machine Learning
- Internet of Things
- Integration

Featured

- Azure Data Explorer — Learn more
- HDInsight — Quickstart tutorial
- Data Lake Analytics — Quickstart tutorial
- Stream Analytics job — Quickstart tutorial
- Analysis Services — Quickstart tutorial
- Azure Databricks — Quickstart tutorial

Start Databricks, Azure



Azure Databricks Service

* Workspace name

db1 ✓

* Subscription

DI_NA_SmartSpaces_001

* Resource group

○ Create new ● Use existing

chc_x_001

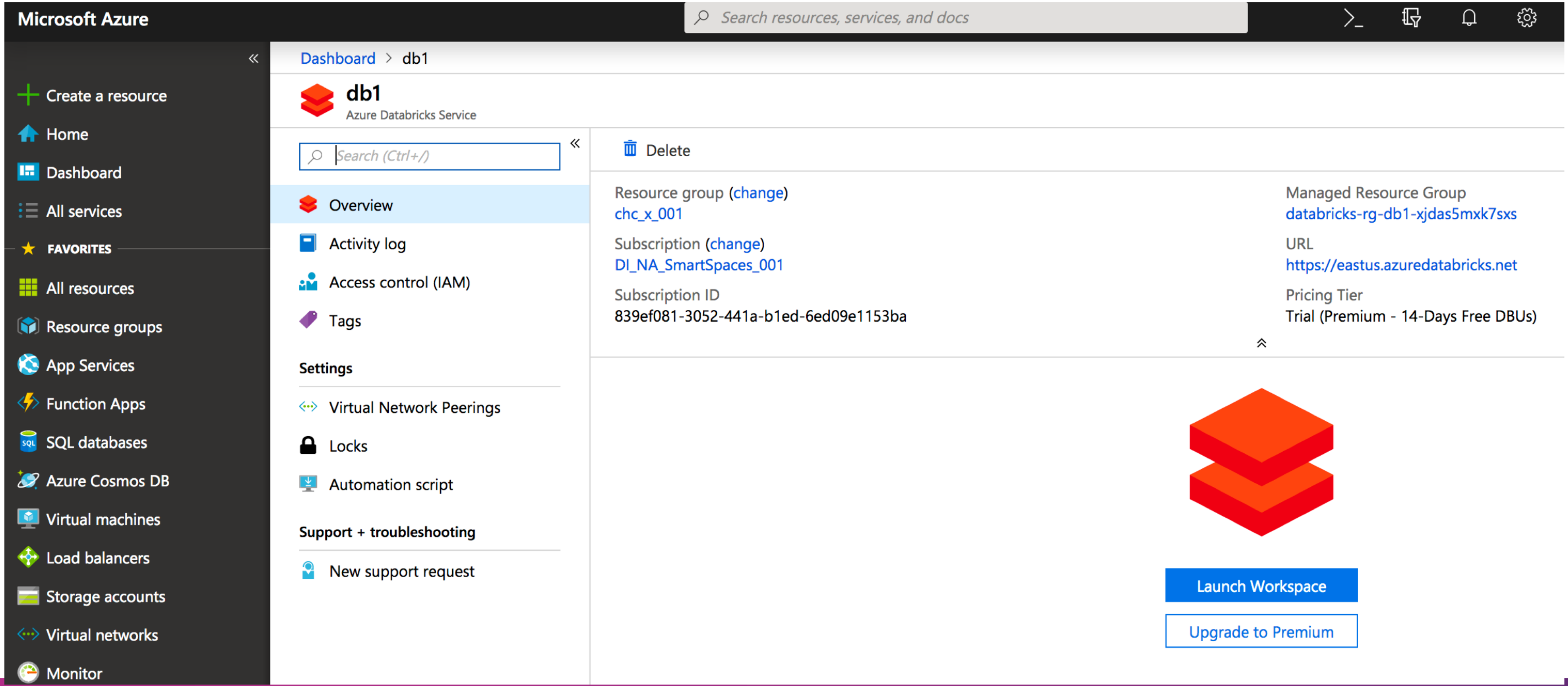
* Location

East US

* Pricing Tier (View full pricing details)

Trial (Premium - 14-Days Free DBUs)

Start Databricks, Azure



Resource group (change)
chc_x_001

Subscription (change)
DI_NA_SmartSpaces_001

Subscription ID
839ef081-3052-441a-b1ed-6ed09e1153ba

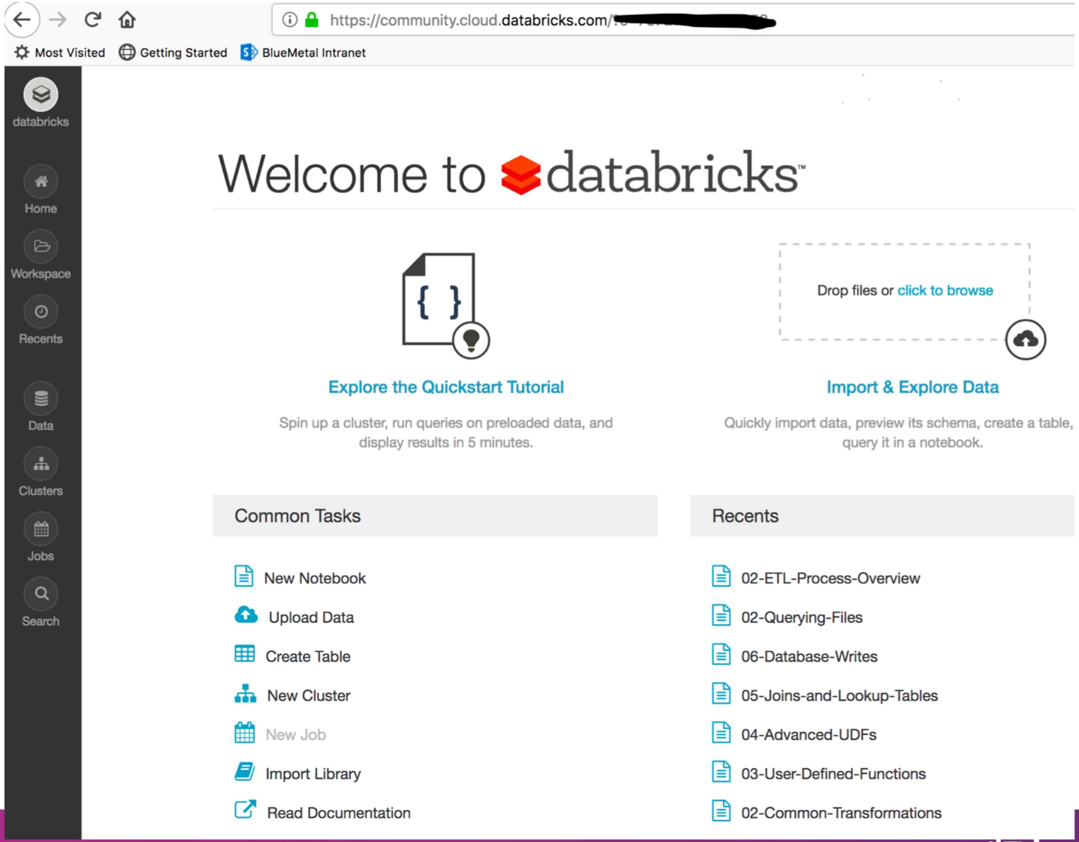
Managed Resource Group
databricks-rg-db1-xjdas5mxk7sxs

URL
https://eastus.azuredatabricks.net

Pricing Tier
Trial (Premium - 14-Days Free DBUs)

Start Databricks, Community

Begin live demo….



https://community.cloud.databricks.com/[redacted]

Welcome to databricks™

Drop files or click to browse

Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Import & Explore Data

Quickly import data, preview its schema, create a table, query it in a notebook.

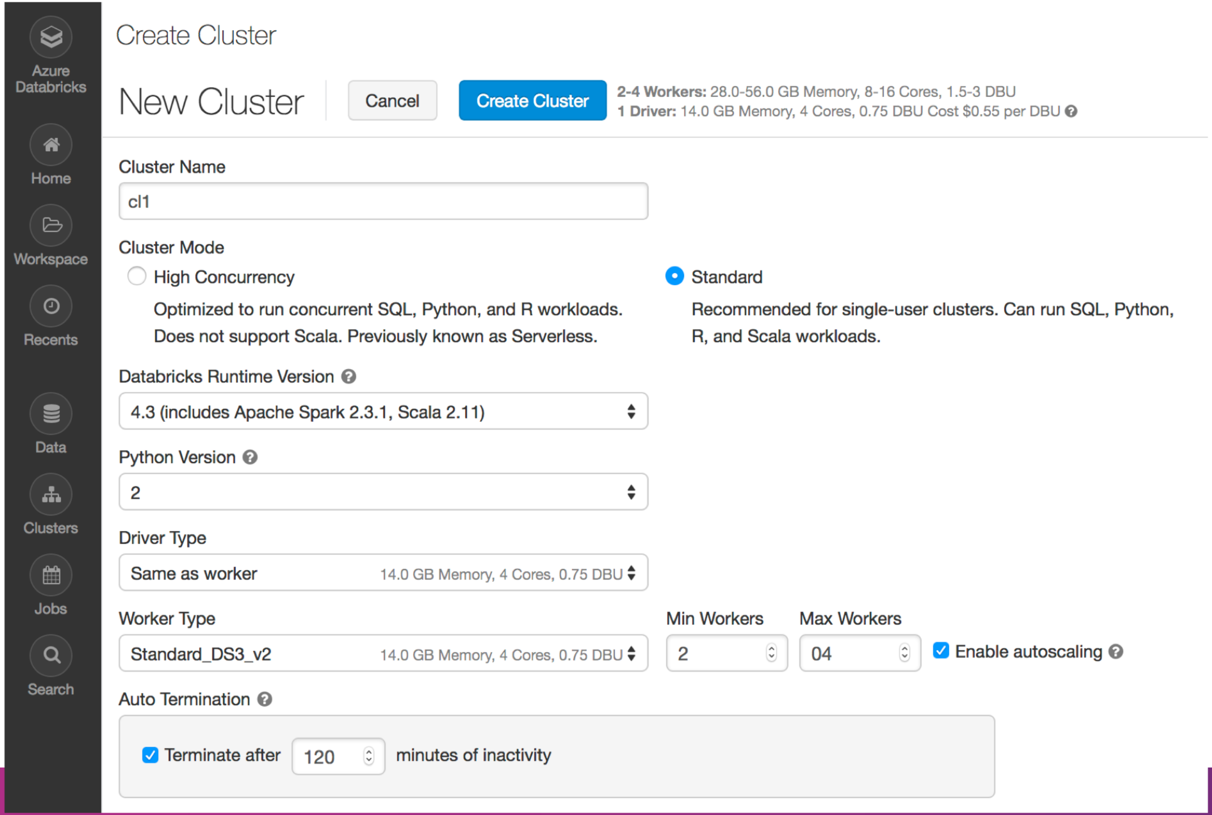
Common Tasks

- New Notebook
- Upload Data
- Create Table
- New Cluster
- New Job
- Import Library
- Read Documentation

Recents

- 02-ETL-Process-Overview
- 02-Querying-Files
- 06-Database-Writes
- 05-Joins-and-Lookup-Tables
- 04-Advanced-UDFs
- 03-User-Defined-Functions
- 02-Common-Transformations

Make a Cluster



Create Cluster

New Cluster

2-4 Workers: 28.0-56.0 GB Memory, 8-16 Cores, 1.5-3 DBU
1 Driver: 14.0 GB Memory, 4 Cores, 0.75 DBU Cost $0.55 per DBU

Cluster Name: cl1

Cluster Mode

○ High Concurrency
Optimized to run concurrent SQL, Python, and R workloads. Does not support Scala. Previously known as Serverless.

● Standard
Recommended for single-user clusters. Can run SQL, Python, R, and Scala workloads.

Databricks Runtime Version: 4.3 (includes Apache Spark 2.3.1, Scala 2.11)

Python Version: 2

Driver Type: Same as worker — 14.0 GB Memory, 4 Cores, 0.75 DBU

Worker Type: Standard_DS3_v2 — 14.0 GB Memory, 4 Cores, 0.75 DBU

Min Workers: 2

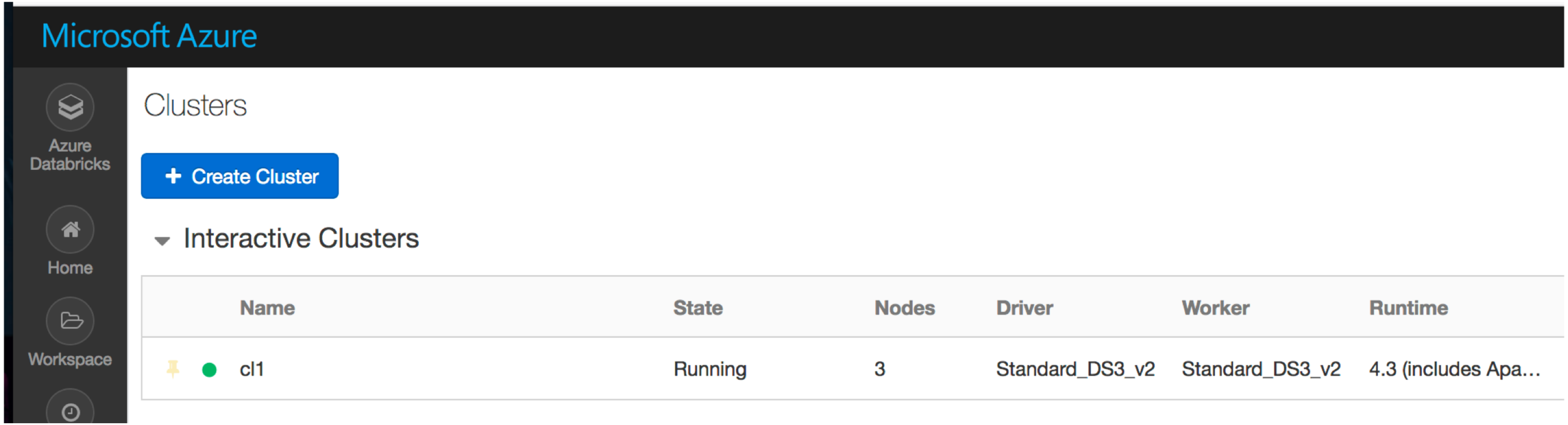
Max Workers: 04

☑ Enable autoscaling

Auto Termination

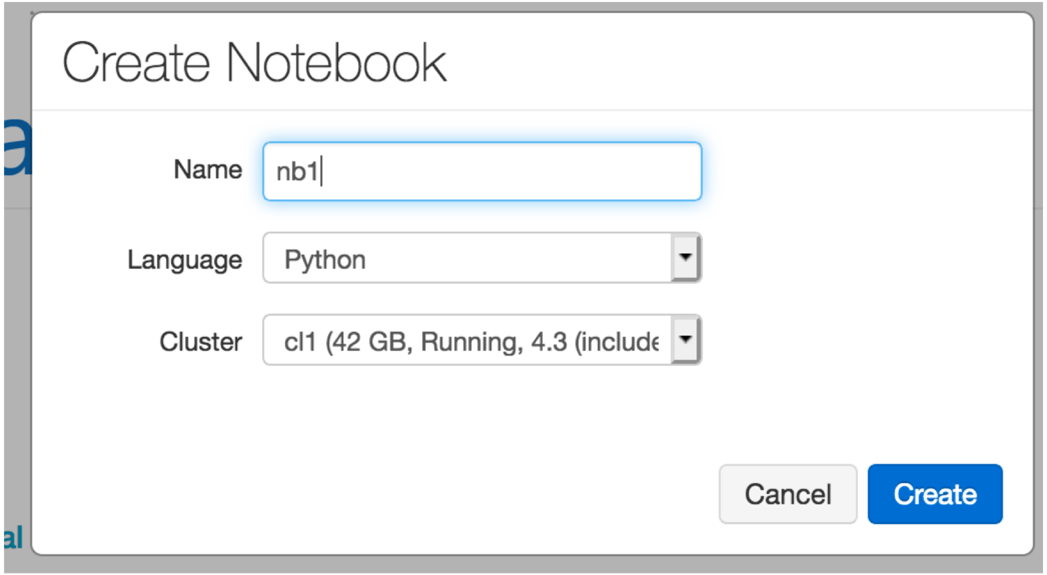
☑ Terminate after 120 minutes of inactivity

Cluster Running



	Name	State	Nodes	Driver	Worker	Runtime
●	cl1	Running	3	Standard_DS3_v2	Standard_DS3_v2	4.3 (includes Apa…

Create/Attach a Notebook



Create Notebook

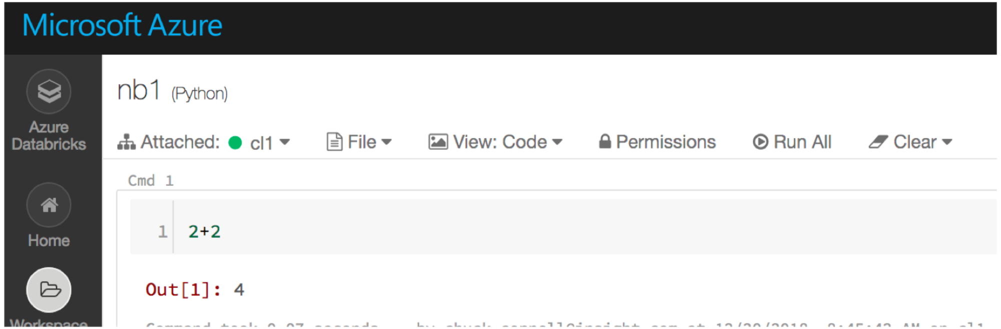
Name: nb1

Language: Python

Cluster: cl1 (42 GB, Running, 4.3 (include

Verify Databricks Resource

- That’s it!
- You have a fully running, auto-scaling Spark cluster
  - Latest (synced) software releases, well tuned
  - Friendly GUI, in your favorite language!
- Has anyone done this same operation manually for Spark?
  - How long did it take?
  - What are the complexities?



Cmd 1

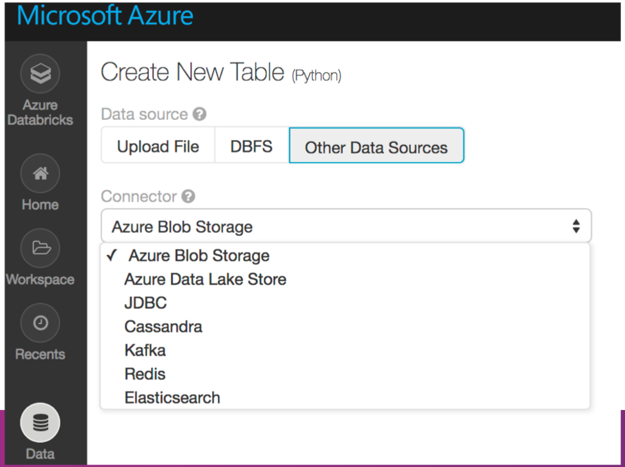
```
2+2
```

```
Out[1]: 4
```

Command took 0.07 seconds -- by chuck.connell@insight.com at 12/20/2018, 9:45:43 AM on cl1

Data Import

- Persistent tables, stored in Databricks File System (DBFS)
  - Backed by Azure Blob
  - Cached in Databricks memory as needed
- We use CSV crime data from data.boston.gov, but many possible sources…



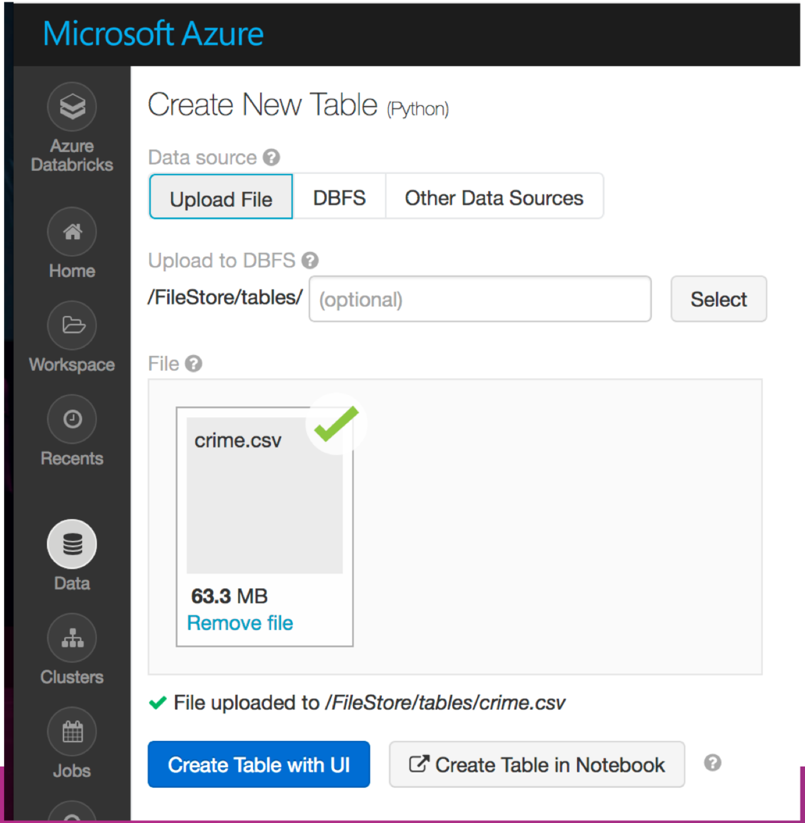
Microsoft Azure

Create New Table (Python)

Data source: Other Data Sources

Connector: Azure Blob Storage

Create Tables



Microsoft Azure

Create New Table (Python)

Data source

Upload File | DBFS | Other Data Sources

Upload to DBFS

/FileStore/tables/ (optional) Select

File

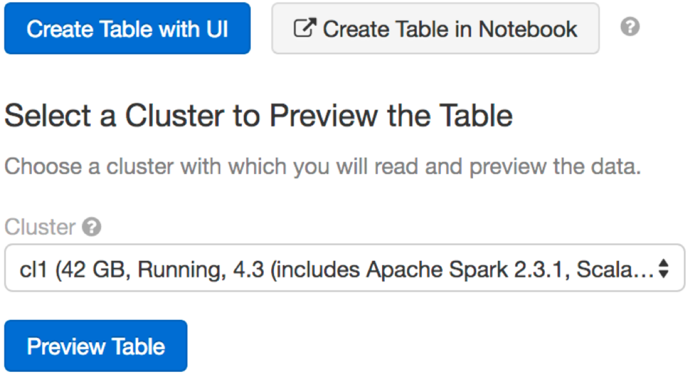
crime.csv

63.3 MB

Remove file

✓ File uploaded to */FileStore/tables/crime.csv*

Create Table with UI | Create Table in Notebook



Create Table with UI | Create Table in Notebook

Select a Cluster to Preview the Table

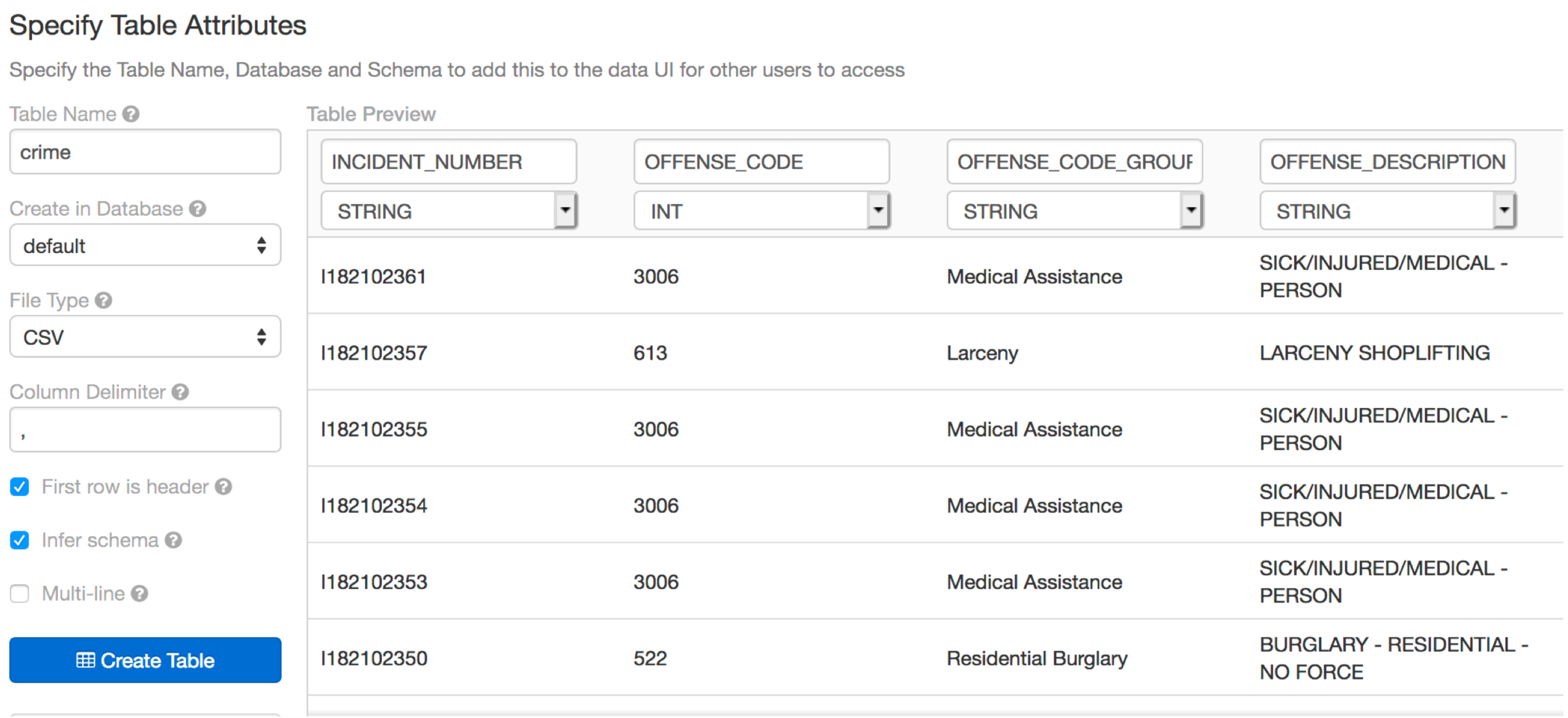
Choose a cluster with which you will read and preview the data.

Cluster

cl1 (42 GB, Running, 4.3 (includes Apache Spark 2.3.1, Scala...

Preview Table

Create Tables



Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: crime

Create in Database: default

File Type: CSV

Column Delimiter: ,

☑ First row is header

☑ Infer schema

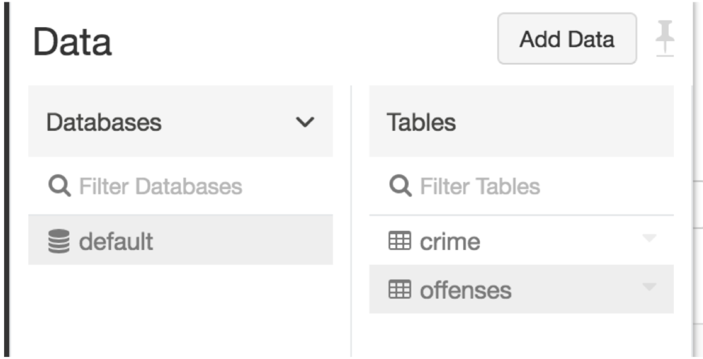
□ Multi-line

Create Table

Table Preview

INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUF	OFFENSE_DESCRIPTION
STRING	INT	STRING	STRING
I182102361	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON
I182102357	613	Larceny	LARCENY SHOPLIFTING
I182102355	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON
I182102354	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON
I182102353	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON
I182102350	522	Residential Burglary	BURGLARY - RESIDENTIAL - NO FORCE

Verify Tables



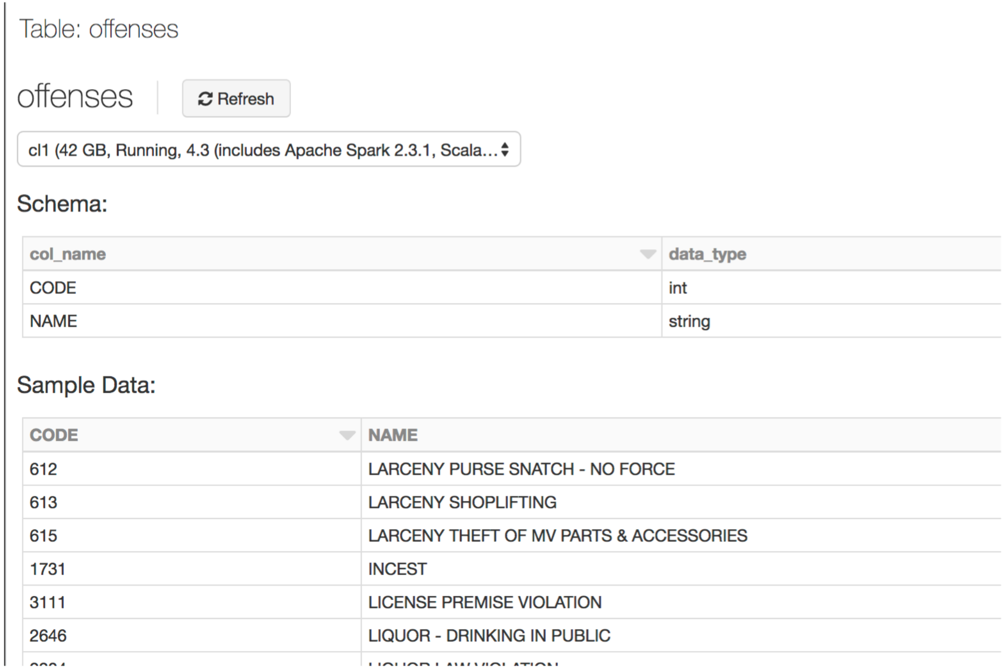


Table: offenses

offenses

cl1 (42 GB, Running, 4.3 (includes Apache Spark 2.3.1, Scala…

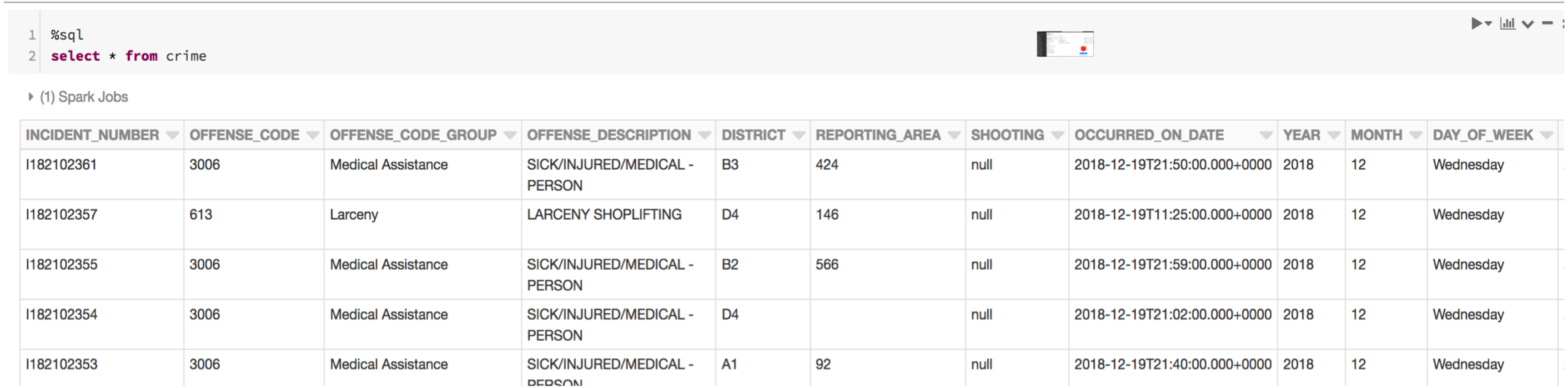
Schema:

col_name	data_type
CODE	int
NAME	string

Sample Data:

CODE	NAME
612	LARCENY PURSE SNATCH - NO FORCE
613	LARCENY SHOPLIFTING
615	LARCENY THEFT OF MV PARTS & ACCESSORIES
1731	INCEST
3111	LICENSE PREMISE VIOLATION
2646	LIQUOR - DRINKING IN PUBLIC

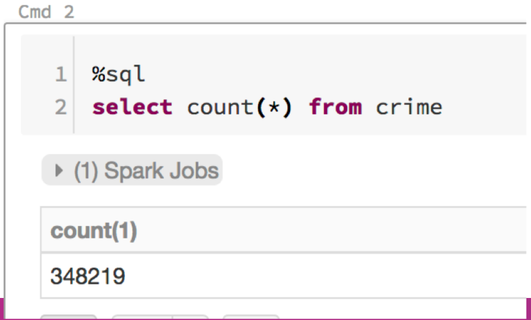
Standard SQL



```
%sql
select * from crime
```

(1) Spark Jobs

INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTING_AREA	SHOOTING	OCCURRED_ON_DATE	YEAR	MONTH	DAY_OF_WEEK
I182102361	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON	B3	424	null	2018-12-19T21:50:00.000+0000	2018	12	Wednesday
I182102357	613	Larceny	LARCENY SHOPLIFTING	D4	146	null	2018-12-19T11:25:00.000+0000	2018	12	Wednesday
I182102355	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON	B2	566	null	2018-12-19T21:59:00.000+0000	2018	12	Wednesday
I182102354	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON	D4		null	2018-12-19T21:02:00.000+0000	2018	12	Wednesday
I182102353	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON	A1	92	null	2018-12-19T21:40:00.000+0000	2018	12	Wednesday



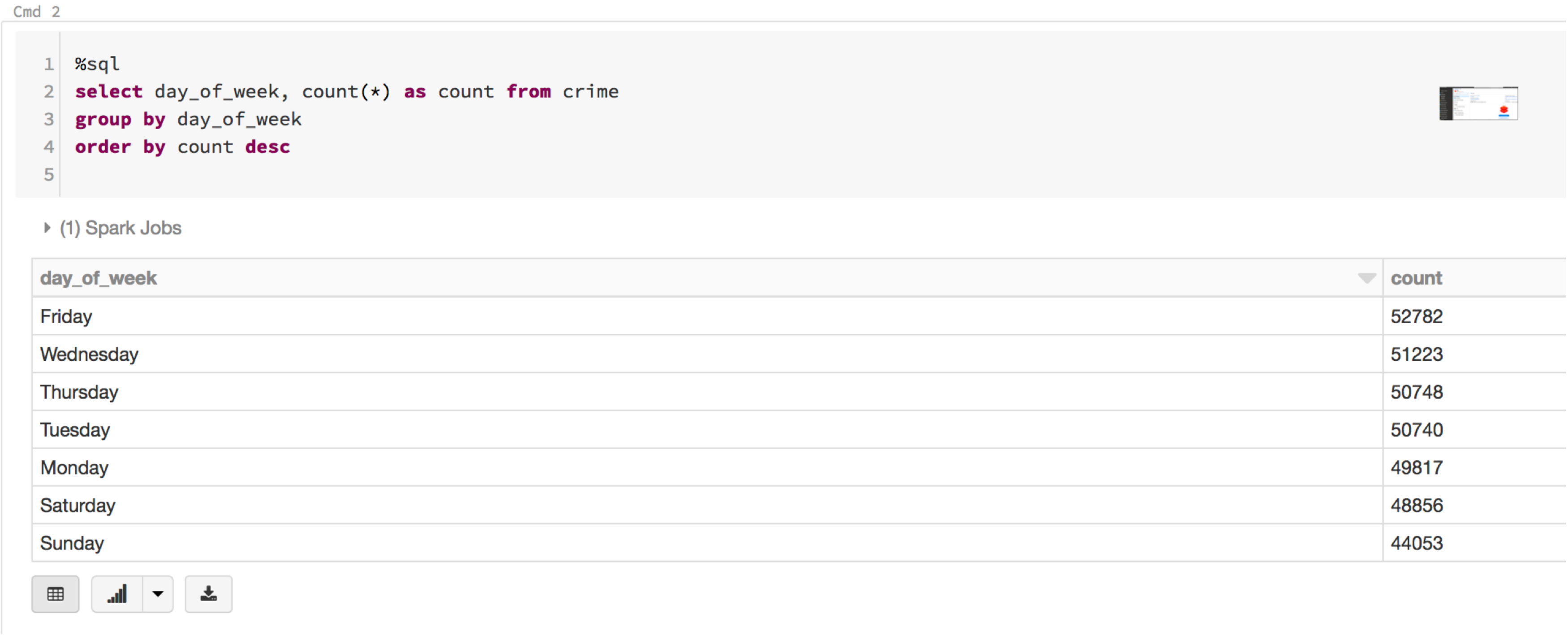
Cmd 2

```
%sql
select count(*) from crime
```

(1) Spark Jobs

count(1)
348219

Standard SQL



Cmd 2

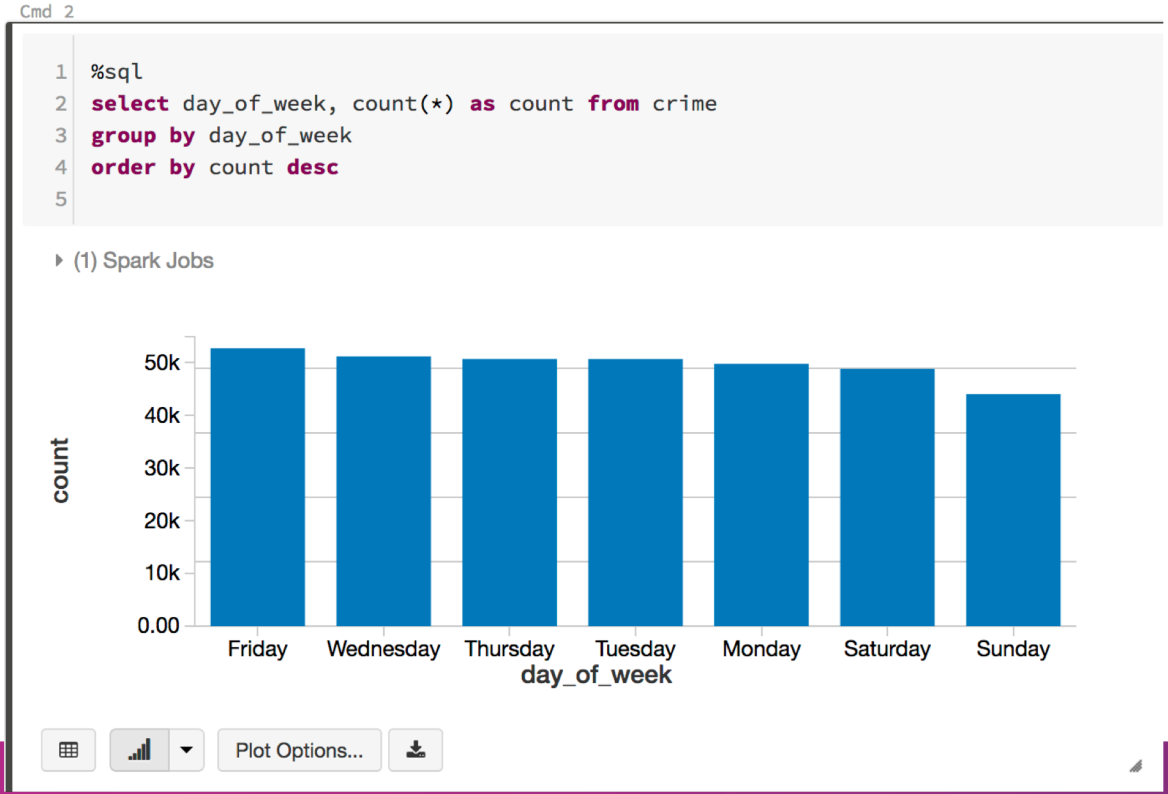
```
%sql
select day_of_week, count(*) as count from crime
group by day_of_week
order by count desc

```

▸ (1) Spark Jobs

day_of_week	count
Friday	52782
Wednesday	51223
Thursday	50748
Tuesday	50740
Monday	49817
Saturday	48856
Sunday	44053

Visualize Data



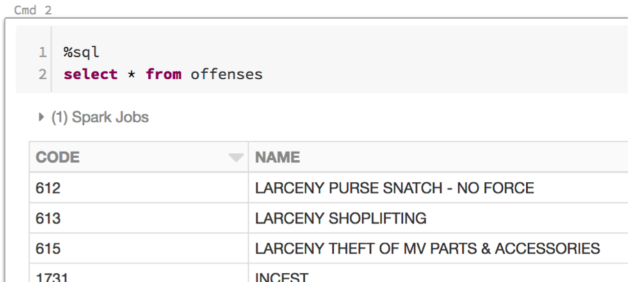
Cmd 2

```
%sql
select day_of_week, count(*) as count from crime
group by day_of_week
order by count desc

```

(1) Spark Jobs

Join Tables

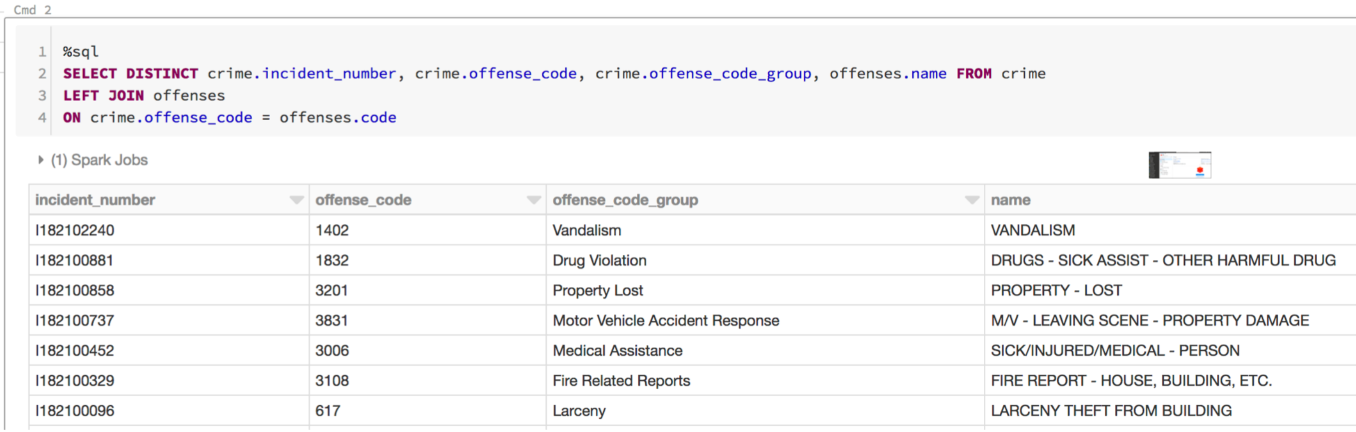


Cmd 2

```
%sql
select * from offenses
```

(1) Spark Jobs

CODE	NAME
612	LARCENY PURSE SNATCH - NO FORCE
613	LARCENY SHOPLIFTING
615	LARCENY THEFT OF MV PARTS & ACCESSORIES
1731	INCEST
3111	LICENSE PREMISE VIOLATION
2646	LIQUOR - DRINKING IN PUBLIC



Cmd 2

```
%sql
SELECT DISTINCT crime.incident_number, crime.offense_code, crime.offense_code_group, offenses.name FROM crime
LEFT JOIN offenses
ON crime.offense_code = offenses.code
```

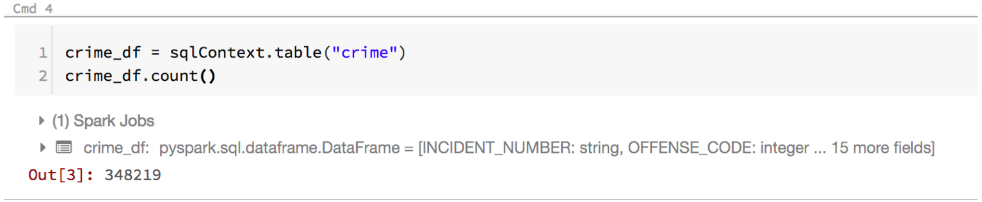
(1) Spark Jobs

incident_number	offense_code	offense_code_group	name
I182102240	1402	Vandalism	VANDALISM
I182100881	1832	Drug Violation	DRUGS - SICK ASSIST - OTHER HARMFUL DRUG
I182100858	3201	Property Lost	PROPERTY - LOST
I182100737	3831	Motor Vehicle Accident Response	M/V - LEAVING SCENE - PROPERTY DAMAGE
I182100452	3006	Medical Assistance	SICK/INJURED/MEDICAL - PERSON
I182100329	3108	Fire Related Reports	FIRE REPORT - HOUSE, BUILDING, ETC.
I182100096	617	Larceny	LARCENY THEFT FROM BUILDING

DataFrame vs Table

- DataFrame is the default data abstraction
- Stored in Spark runtime memory
  - When needed, can be persisted to DBFS (Parquet serialization is default) or SQL table
- DataFrame operations a super-set of SQL, also includes
  - ETL -- dropna, na.fill, explode, moved mal-formed elsewhere
  - Convert to GraphFrame
  - Submit to ML libraries

DataFrame Example



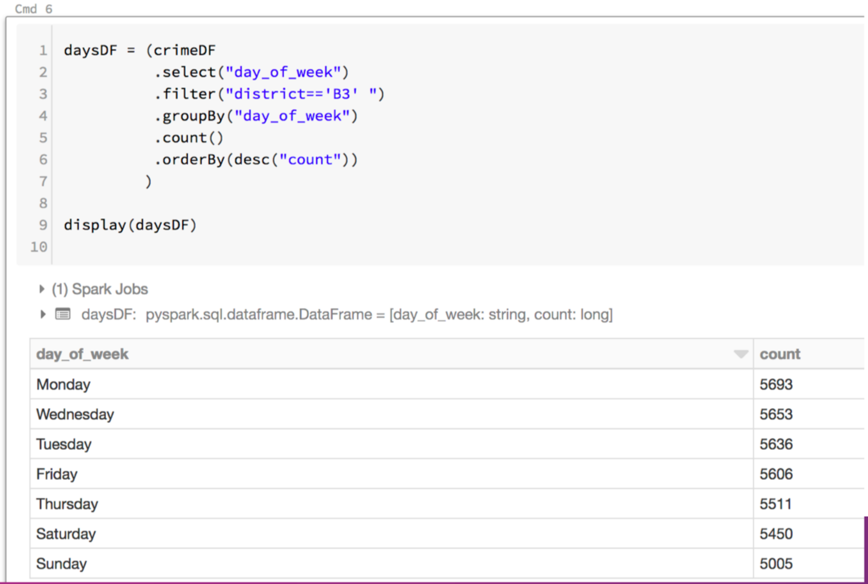
Cmd 4

```
crime_df = sqlContext.table("crime")
crime_df.count()
```

▸ (1) Spark Jobs

▸ crime_df: pyspark.sql.dataframe.DataFrame = [INCIDENT_NUMBER: string, OFFENSE_CODE: integer ... 15 more fields]

```
Out[3]: 348219
```



Cmd 6

```
daysDF = (crimeDF
          .select("day_of_week")
          .filter("district=='B3' ")
          .groupBy("day_of_week")
          .count()
          .orderBy(desc("count"))
         )

display(daysDF)

```

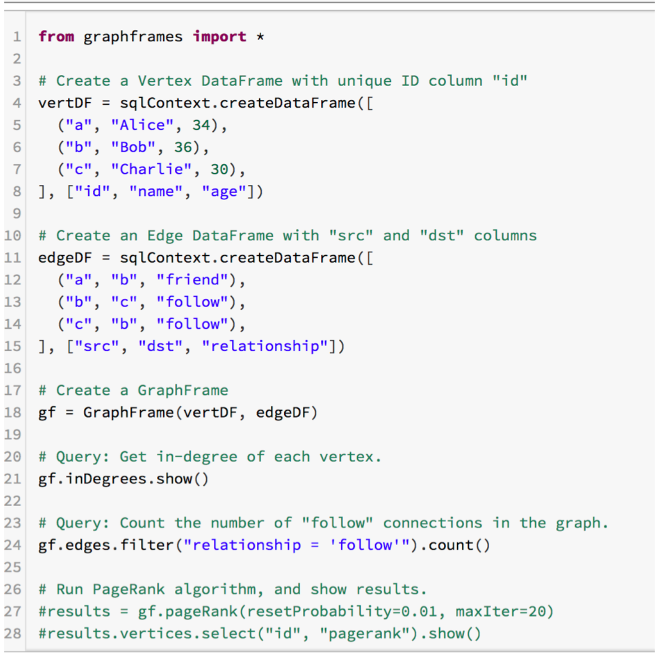
▸ (1) Spark Jobs

▸ daysDF: pyspark.sql.dataframe.DataFrame = [day_of_week: string, count: long]

day_of_week	count
Monday	5693
Wednesday	5653
Tuesday	5636
Friday	5606
Thursday	5511
Saturday	5450
Sunday	5005

Import Library / GraphFrames

- Databricks Home / Import Library / Maven / Search / Spark Packages / graph
    - graphframes:graphframes:0.6.0-spark2.3-s_2.11



```python
from graphframes import *

# Create a Vertex DataFrame with unique ID column "id"
vertDF = sqlContext.createDataFrame([
  ("a", "Alice", 34),
  ("b", "Bob", 36),
  ("c", "Charlie", 30),
], ["id", "name", "age"])

# Create an Edge DataFrame with "src" and "dst" columns
edgeDF = sqlContext.createDataFrame([
  ("a", "b", "friend"),
  ("b", "c", "follow"),
  ("c", "b", "follow"),
], ["src", "dst", "relationship"])

# Create a GraphFrame
gf = GraphFrame(vertDF, edgeDF)

# Query: Get in-degree of each vertex.
gf.inDegrees.show()

# Query: Count the number of "follow" connections in the graph.
gf.edges.filter("relationship = 'follow'").count()

# Run PageRank algorithm, and show results.
#results = gf.pageRank(resetProbability=0.01, maxIter=20)
#results.vertices.select("id", "pagerank").show()
```

Databricks Connections

- Getting data in
  - CSV, JSON, Parquet, LZO, Zip, Avro
  - Hive tables
  - Azure Blob or Data Lake as DBFS directory
  - Any RDBMS with JDBC
  - Azure Data Hub, which has *many* source connectors
- Getting data out
  - Write to many file formats
  - JBDC and ODBC for programmatic inbound reads
- REST API
  - Clusters, DBFS, jobs, libraries, workspaces…

Databricks Goodies

- Databricks Delta, ACID compliant transactions
- Security integration with Azure Active Directory
  - See my article on LinkedIn for details
- GraphFrames
  - A library of routines for creating and calculating node/edge data structures
  - Ex: shortest path, PageRank
- Machine Learning
  - A library and workflow for many common ML techniques
  - Support for many third-party ML libs – H2O, scikit-learn, DataRobot, XGBoost
- R language

Caveats

- No option for local install, so no “hybrid cloud” option
  - Databricks not in Azure Stack (afaik)
- Spark/Databricks relatively slow for small data sets
  - Key-value stores (Redis, Couchbase) have <1ms response
  - RDBMS have few ms response for tuned SQL queries
  - Fastest Spark query is ~400ms
  - Interesting tradeoffs for specific use-cases (1M vs 1T rows)
- Overall "fit and finish" within Azure
  - Control of allocation within resource groups
  - Programmatic creation of base Azure Databricks resource for DevOps CI/CD.

Next Steps

- https://databricks.com/spark/comparing-databricks-to-apache-spark (Databricks vs Spark)
- http://community.cloud.databricks.com (Community edition)
- https://azure.microsoft.com/en-us/services/databricks (Azure Databricks)
- https://academy.databricks.com (Databricks training)
- https://docs.databricks.com/spark/latest/mllib/index.html (Machine learning)
- https://docs.databricks.com/spark/latest/graph-analysis/graphframes/index.html (GraphFrames)
- https://docs.databricks.com/api/latest/index.html (REST API)

Questions / Discussion…. ??

Insight | Digital Innovation

Thank You