DATA+AI SUMMIT BY databricks

10x Easier Development Using Databricks Connect

Martin Grund
Principal Software Engineer, Databricks

Vladislav Mantic-Lugo
Product Manager, Databricks

©2024 Databricks Inc. — All rights reserved

Product safe harbor statement

This information is provided to outline Databricks' general product direction and is for informational purposes only. Customers who purchase Databricks services should make their purchase decisions relying solely upon services, features, and functions that are currently available. Unreleased features or functionality described in forward-looking statements are subject to change at Databricks discretion and may not be delivered as planned or at all

Who prefers to develop in:

1) a Notebook?
2) an IDE?

And why?

©2024 Databricks Inc. — All rights reserved

IDE-BASED DEVELOPMENT W/ DATABRICKS

Why do Databricks users love developing in an IDE?

- Meet developers where they are… muscle memory!
- Software engineering best practises: debugging, code modularization, unit testing, etc.

Why users love DB Connect:

- With any IDE,
- Interactive debugging of Spark code on Databricks cluster,
- High production fidelity: debug & run on target platform,
- Application development: Use DB Connect as a “Python-native JDBC driver”.

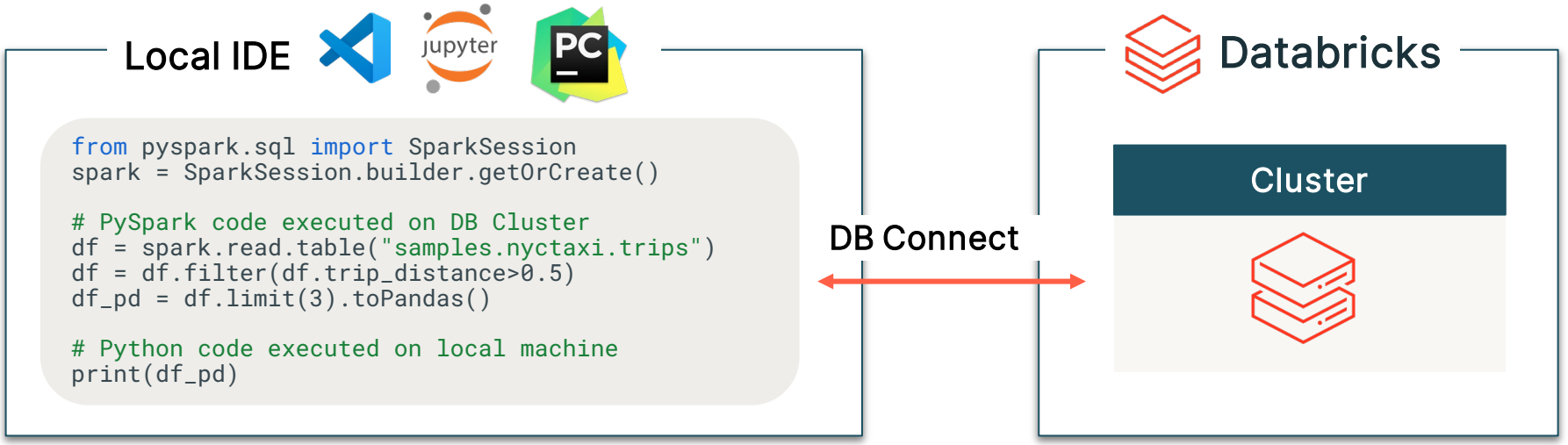
©2024 Databricks Inc. — All rights reserved

WALK-THROUGH

HOW TO USE DB CONNECT AND PYCHARM

©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

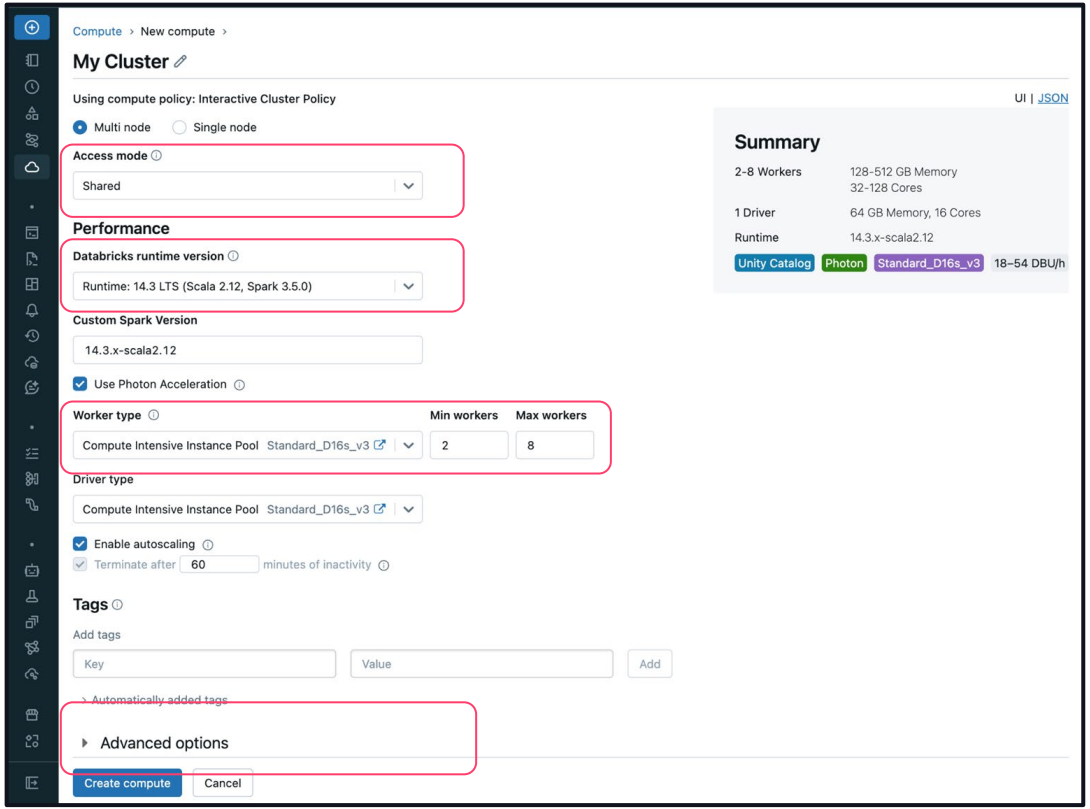


©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

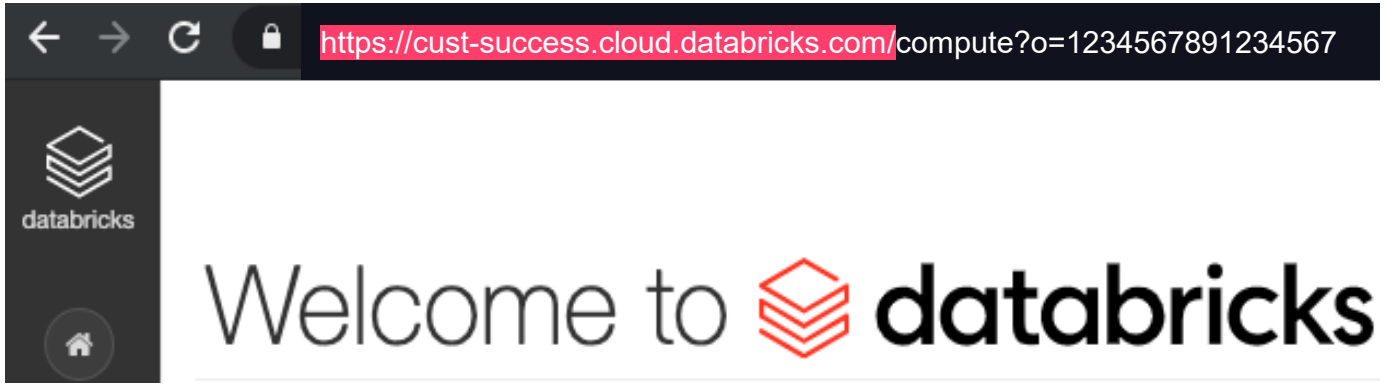
> 1. I create a cluster



DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

✓ 1. I create a cluster
> 2. I copy my workspace URL



©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

✓ 1. I create a cluster
✓ 2. I copy my workspace URL
> 3. Install Python package. `pip install databricks-connect==14.3`

```
$ python3.10 -m venv ./venv
$ source ./venv/bin/activate
(venv)$ pip3 install --upgrade "databricks-connect==14.3.*"
```


©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

✓ 1. I create a cluster
✓ 2. I copy my workspace URL
✓ 3. Install Python package, `pip install databricks-connect==14.3`
> 4. Create a config profile (workspace & cluster ID, auth token)

```
$ brew install databricks
$ databricks auth login --configure-cluster --host https://cust-success.cloud.databricks.com
```

- Choose profile name: DEFAULT
- Authenticate with OAuth
- Select cluster -> cluster ID

©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

✓ 1. I create a cluster
✓ 2. I copy my workspace URL
✓ 3. Install Python package, `pip install databricks-connect==14.3`
✓ 4. Create a config profile (workspace & cluster ID, auth token)

```
$ brew install databricks
$ databricks auth login --configure-cluster --host https://cust-success.cloud.databricks.com
```

> 5. Open my IDE and I create my Spark session using a config profile

```
from databricks.connect import DatabricksSession
spark = DatabricksSession.builder.profile("DEFAULT").getOrCreate()
```


©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

Install DB Connect and configure my Spark session

✓ 1. I create a cluster
✓ 2. I copy my workspace URL
✓ 3. Install Python package, `pip install databricks-connect==14.3`
✓ 4. Create a config profile (workspace & cluster ID, auth token)

```
$ brew install databricks
$ databricks auth login --configure-cluster --host https://cust-success.cloud.databricks.com
```

✓ 5. Open my IDE and I create my Spark session using a config profile

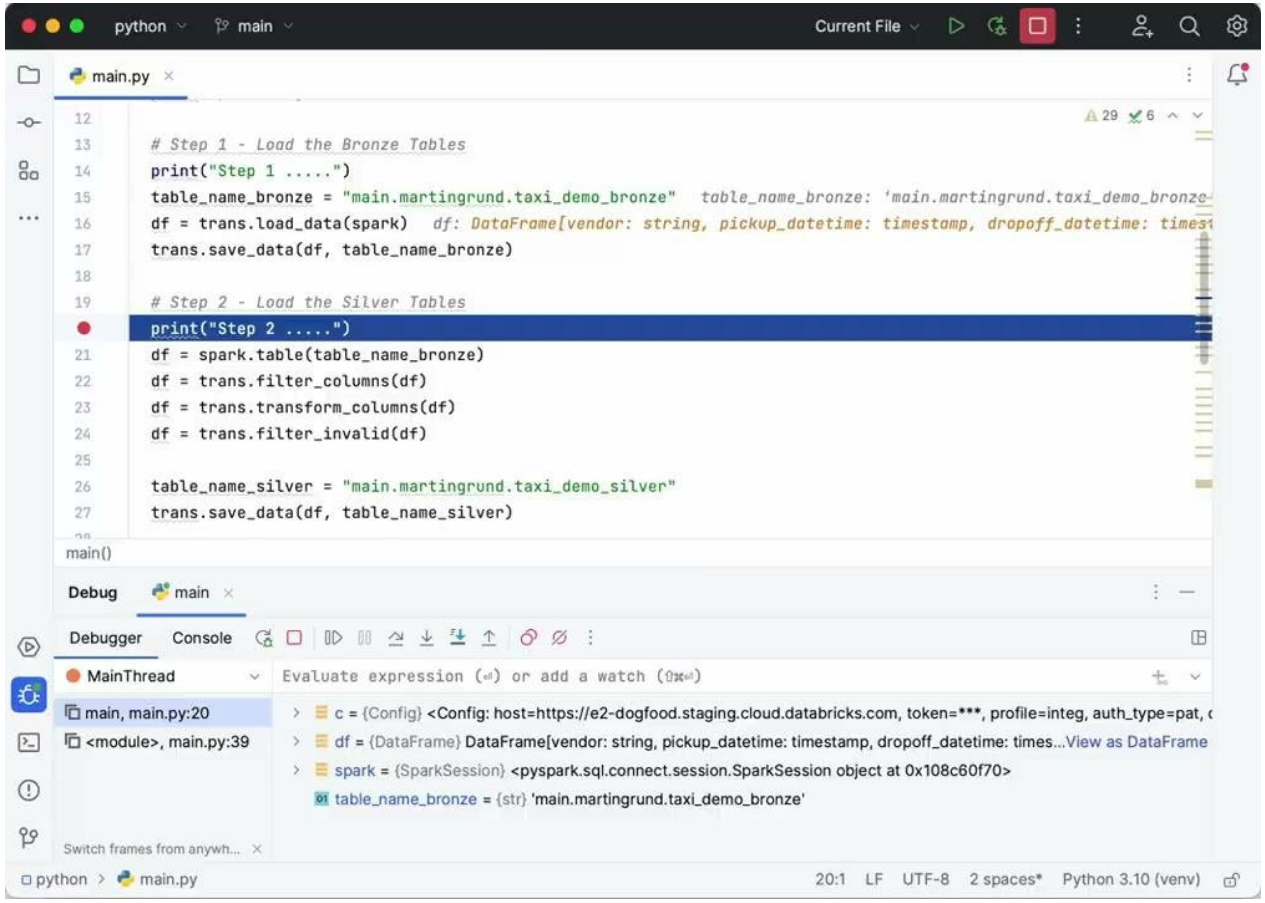
```
from databricks.connect import DatabricksSession
spark = DatabricksSession.builder.profile("DEFAULT").getOrCreate()
```

> 6. Develop, debug and run my *application!*

```
df = spark.read.table("samples.nyctaxi.trips")
df.where(df.trip_distance < 10).show(10)
```


©2024 Databricks Inc. — All rights reserved

DEVELOPMENT FROM THE IDE WITH DB CONNECT

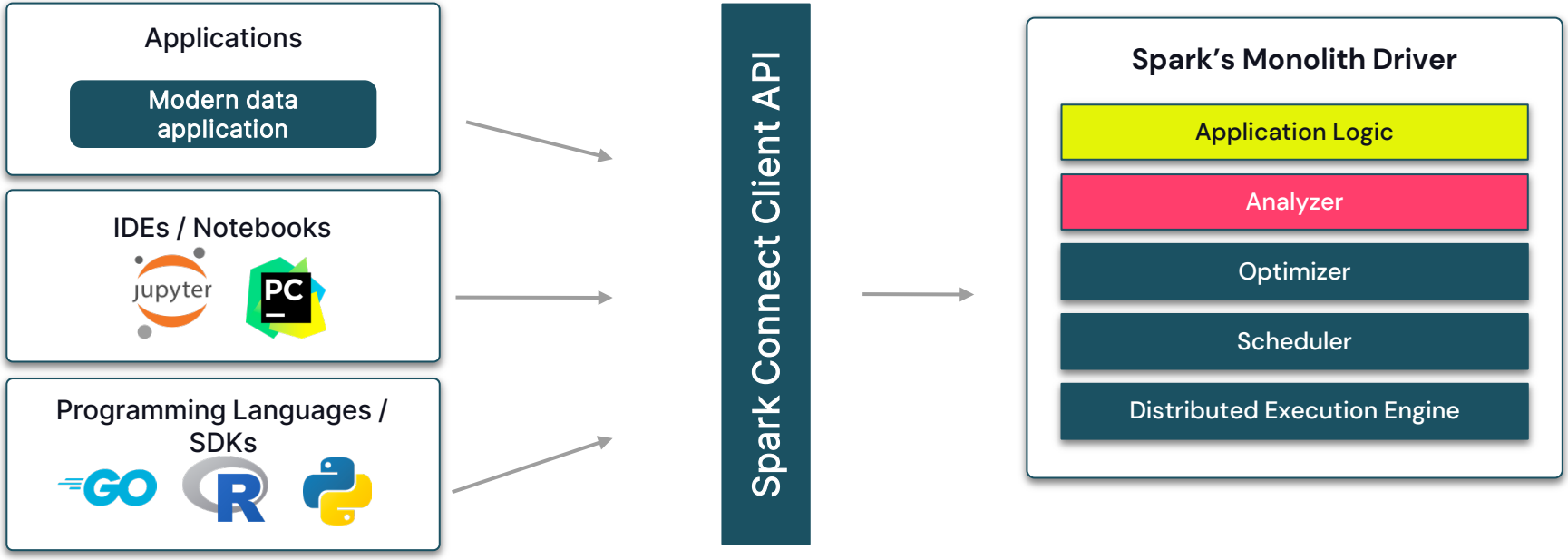


BUT HOW DOES IT WORK

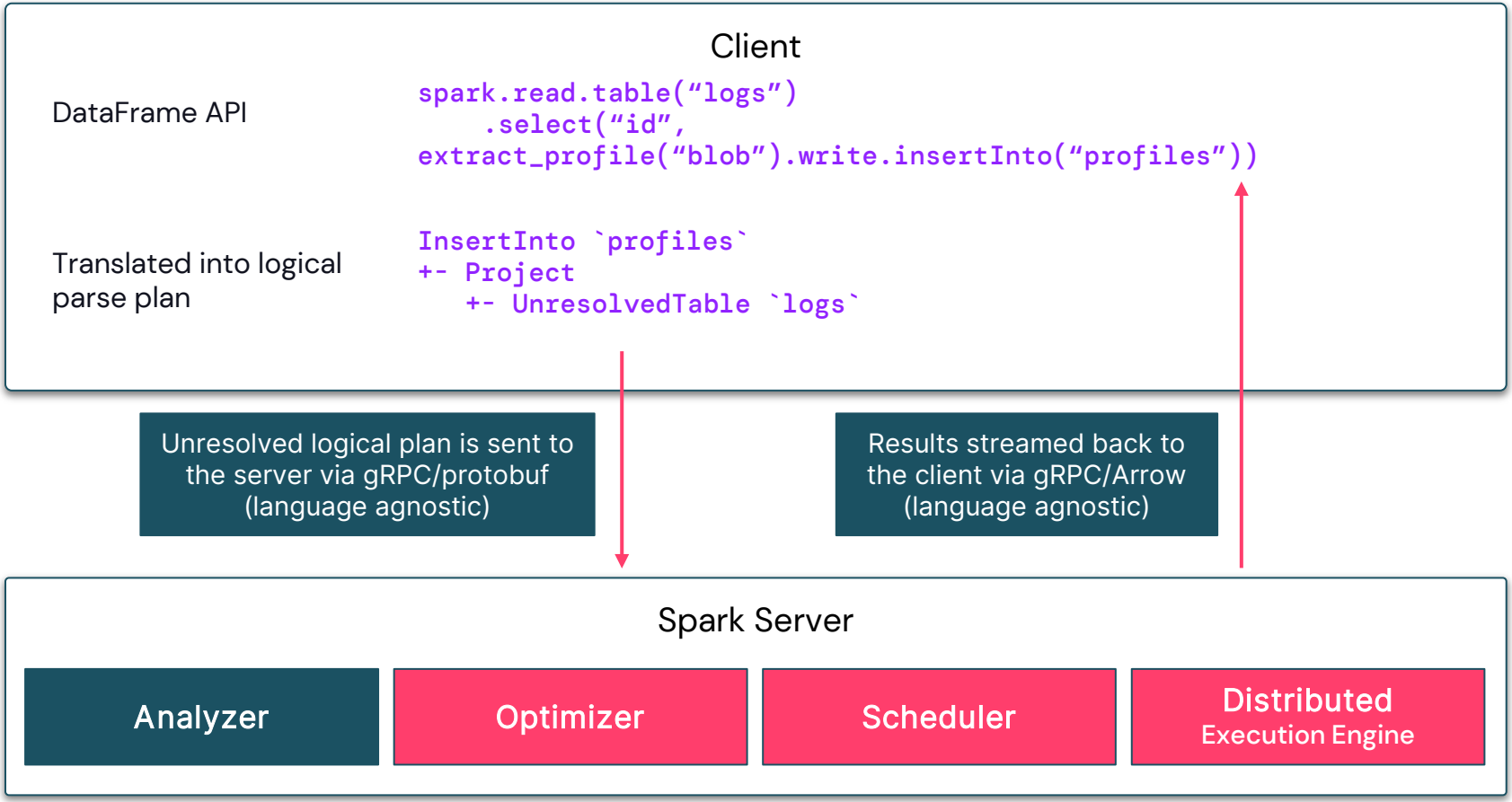
©2024 Databricks Inc. — All rights reserved

SPARK CONNECT

Remote Connectivity: thin client, full power of Apache Spark



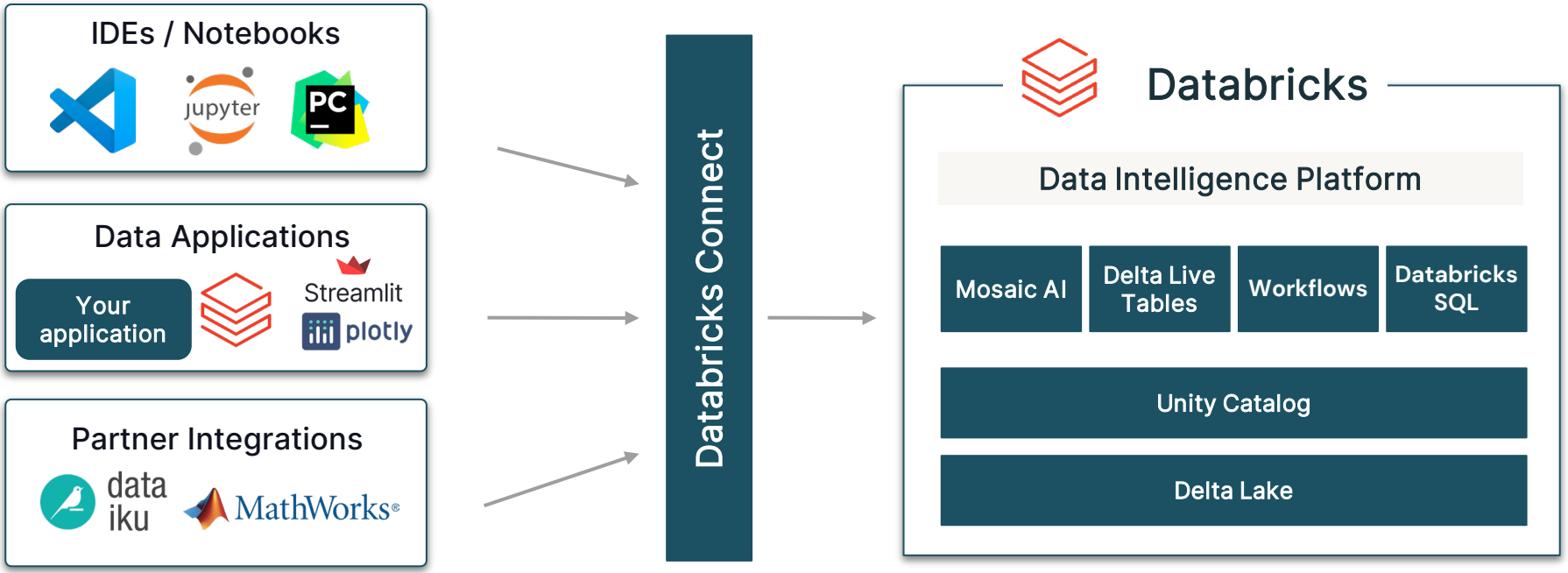
©2024 Databricks Inc. — All rights reserved



©2024 Databricks Inc. — All rights reserved

DATABRICKS CONNECT USES SPARK CONNECT

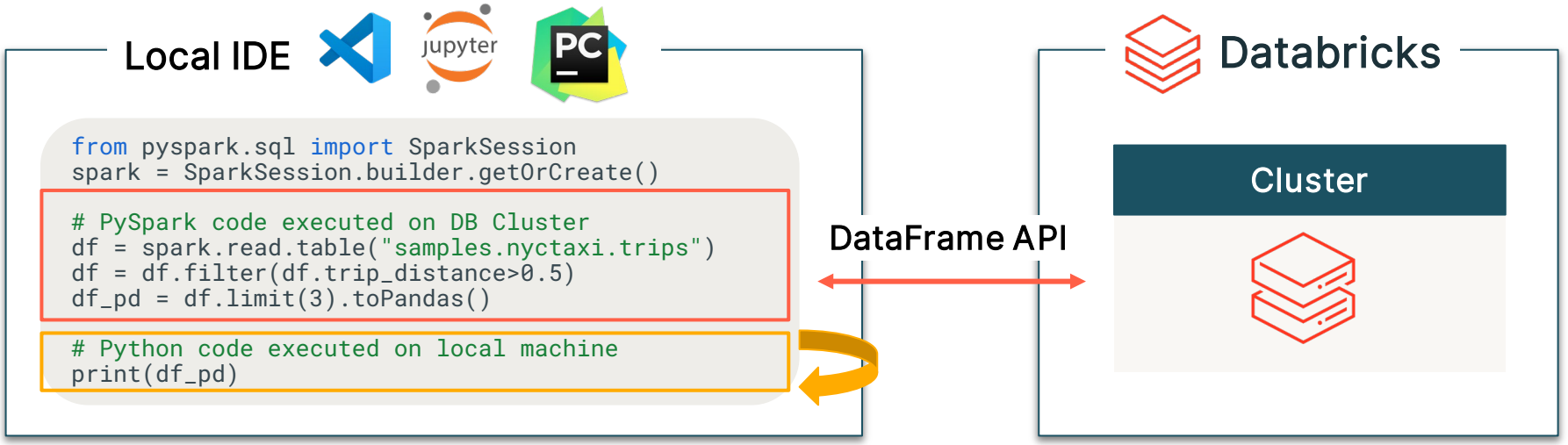
Connect to Databricks from anywhere!



©2024 Databricks Inc. — All rights reserved

HOW DATABRICKS CONNECT WORKS

Build on Spark Connect

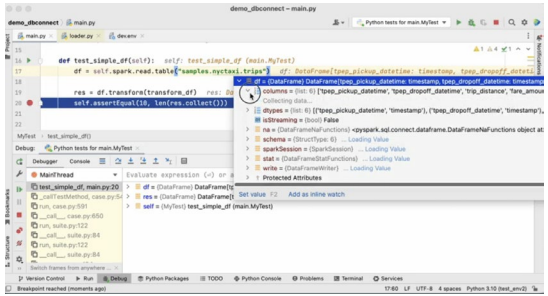


©2024 Databricks Inc. — All rights reserved

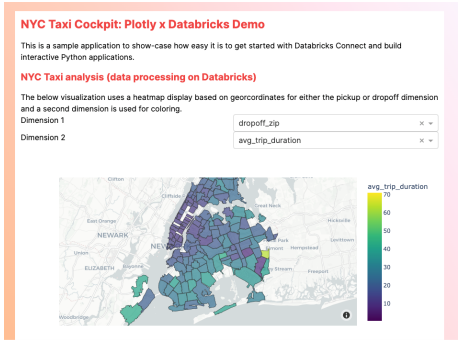
USE DATABRICKS FROM ANYWHERE!

It's more than debugging: Remote Connectivity to Databricks

Interactively develop & debug from IDE



Interactive Data Apps with a few lines of code!



Partner integrations made easy!

Dataframe API (Python, Scala, R)
UDFs, Streaming
ETL github example

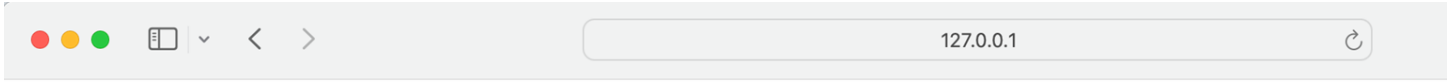
Databricks Lakhouse APPs!

Plotly app github example

Dataiku Demo

©2024 Databricks Inc. — All rights reserved

YOUR APPS WITH DB CONNECT



	index	tpep_pickup_datetime	tpep_dropoff_datetime	trip_distance	fare_amount	pickup_zip	dropoff_zip
0	0	2016-02-16 22:40:45	2016-02-16 22:59:25	5.35	18.5	10003	11238
1	1	2016-02-05 16:06:44	2016-02-05 16:26:03	6.50	21.5	10282	10001
2	2	2016-02-08 07:39:25	2016-02-08 07:44:14	0.90	5.5	10119	10003
3	3	2016-02-29 22:25:33	2016-02-29 22:38:09	3.50	13.5	10001	11222
4	4	2016-02-03 17:21:02	2016-02-03 17:23:24	0.30	3.5	10028	10028
5	5	2016-02-10 00:47:44	2016-02-10 00:53:04	0.00	5.0	10038	10005
6	6	2016-02-19 03:24:25	2016-02-19 03:44:56	6.57	21.5	10001	11377
7	7	2016-02-02 14:05:23	2016-02-02 14:23:07	1.08	11.5	10103	10167
8	8	2016-02-20 15:42:20	2016-02-20 15:50:40	0.80	7.0	10003	10011
9	9	2016-02-14 16:19:53	2016-02-14 16:32:10	1.30	9.0	10199	10020
10	10	2016-02-16 21:01:22	2016-02-16 21:11:29	2.74	11.0	10002	11211
11	11	2016-02-22 20:13:11	2016-02-22 20:27:35	2.09	11.5	10012	10009
12	12	2016-02-12 11:35:59	2016-02-12 11:50:53	2.80	13.5	10103	10016
13	13	2016-02-27 07:50:33	2016-02-27 07:56:38	1.00	6.0	10022	10110
14	14	2016-02-28 12:04:09	2016-02-28 12:07:01	0.70	4.5	10065	10021
15	16	2016-02-11 12:44:24	2016-02-11 13:07:43	3.80	18.0	10021	10027
16	17	2016-02-28 18:25:18	2016-02-28 18:31:08	0.98	6.0	10119	10009
17	18	2016-02-13 20:00:15	2016-02-13 20:17:20	2.83	13.5	10003	10282
18	19	2016-02-21 01:07:04	2016-02-21 01:09:43	0.60	4.0	10018	10018
19	20	2016-02-27 15:56:56	2016-02-27 16:04:51	1.53	8.0	10023	10025
20	21	2016-02-18 07:40:28	2016-02-18 07:49:00	0.93	7.0	10020	10167

©2024 Databricks Inc. — All rights reserved

YOUR APPS WITH DB CONNECT

Avoid Impedance Mismatch

```
@app.route("/")
def main():
 table = "samples.nyctaxi.trips"
 df = spark.table(table)

 if request.args.get('bounds'):
    filter = request.args.get('bounds')
    df = df.where(df.trip_distance < filter)

 if request.args.get('limit'):
    df = df.limit(int(request.args.get("limit")))
 else:
    df = df.limit(100)

 return str(df.toPandas().to_html())
```

```
@app.route("/")
def main():
 con = sqlite3.connect("data.db")
 query  = "select * from trips"

 bounds = None; has_filter = False; params = []
 if request.args.get('bounds'):
    has_filter = True
    query += " where trip_distance < :1"
    params.append(request.args.get('bounds'))

 limit = 100
 if request.args.get('limit'):
    limit = int(request.args.get("limit"))

 query += " limit :2"
 params.append(limit)
 pdf = pd.read_sql(query, con, params=params)
 return str(pdf.to_html())
```


©2024 Databricks Inc. — All rights reserved

YOUR APPS WITH DB CONNECT

Avoid Impedance Mismatch

```
@app.route("/")
def main():
 table = "samples.nyctaxi.trips"
 df = spark.table(table)

 if request.args.get('bounds'):
    filter = request.args.get('bounds')
    df = df.where(df.trip_distance < filter)

 if request.args.get('limit'):
    df = df.limit(int(request.args.get("limit")))
 else:
    df = df.limit(100)

 return str(df.toPandas().to_html())
```

String concatenation!

Significant whitespace!

```
@app.route("/")
def main():
 con = sqlite3.connect("data.db")
 query  = "select * from trips"

 bounds = None; has_filter = False; params = []
 if request.args.get('bounds'):
    has_filter = True
    query += " where trip_distance < :1"
    params.append(request.args.get('bounds'))

 limit = 100
 if request.args.get('limit'):
    limit = int(request.args.get("limit"))

 query += " limit :2"
 params.append(limit)
 pdf = pd.read_sql(query, con, params=params)
 return str(pdf.to_html())
```


©2024 Databricks Inc. — All rights reserved

YOUR APPS WITH DB CONNECT

Debugging!

```python
@app.route("/")
def main():
 table = "samples.nyctaxi.trips"
 df = spark.table(table)

 if request.args.get('bounds'):
    filter = request.args.get('bounds')
    df = df.where(df.trip_distance < filter)

 if request.args.get('limit'):
    df = df.limit(int(request.args.get("limit")))
 else:
    df = df.limit(100)

 return str(df.toPandas().to_html())
```

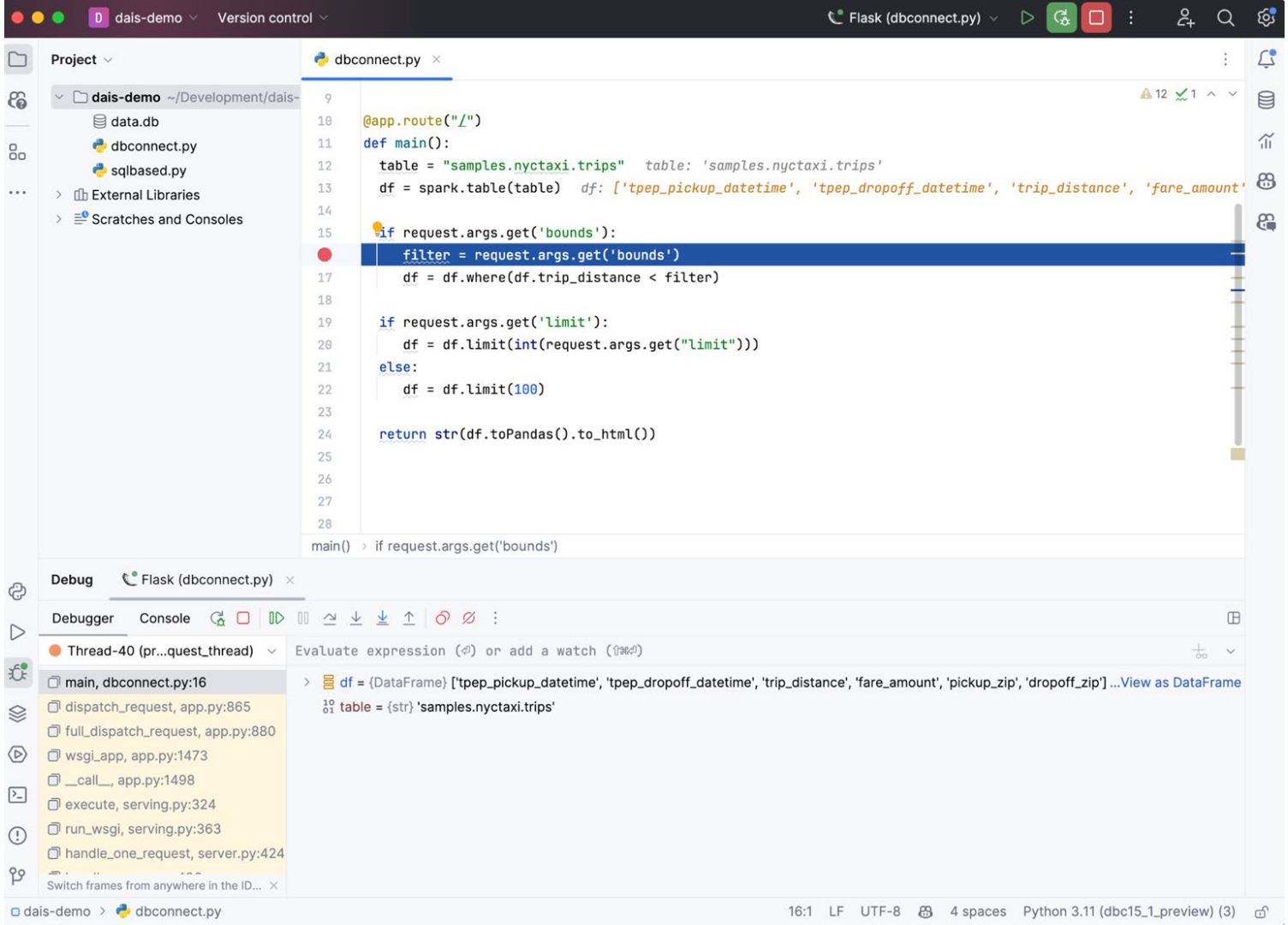
```python
@app.route("/")
def main():
 con = sqlite3.connect("data.db")
 query  = "select * from trips"

 bounds = None; has_filter = False; params = []
 if request.args.get('bounds'):
    has_filter = True
    query += " where trip_distance < :1"
    params.append(request.args.get('bounds'))

 limit = 100
 if request.args.get('limit'):
    limit = int(request.args.get("limit"))

 query += " limit :2"
 params.append(limit)
 pdf = pd.read_sql(query, con, params=params)
 return str(pdf.to_html())
```


©2024 Databricks Inc. — All rights reserved


dais-demo
Version control
Flask (dbconnect.py)
Project
dais-demo ~/Development/dais-
data.db
dbconnect.py
sqlbased.py
External Libraries
Scratches and Consoles
dbconnect.py
@app.route("/")
def main():
table = "samples.nyctaxi.trips" table: 'samples.nyctaxi.trips'
df = spark.table(table) df: ['tpep_pickup_datetime', 'tpep_dropoff_datetime', 'trip_distance', 'fare_amount'
if request.args.get('bounds'):
filter = request.args.get('bounds')
df = df.where(df.trip_distance < filter)
if request.args.get('limit'):
df = df.limit(int(request.args.get("limit")))
else:
df = df.limit(100)
return str(df.toPandas().to_html())
main() › if request.args.get('bounds')
Debug
Flask (dbconnect.py)
Debugger
Console
Thread-40 (pr...quest_thread)
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
main, dbconnect.py:16
dispatch_request, app.py:865
full_dispatch_request, app.py:880
wsgi_app, app.py:1473
__call__, app.py:1498
execute, serving.py:324
run_wsgi, serving.py:363
handle_one_request, server.py:424
Switch frames from anywhere in the ID...
df = {DataFrame} ['tpep_pickup_datetime', 'tpep_dropoff_datetime', 'trip_distance', 'fare_amount', 'pickup_zip', 'dropoff_zip'] ...View as DataFrame
table = {str} 'samples.nyctaxi.trips'
dais-demo › dbconnect.py
16:1
LF
UTF-8
4 spaces
Python 3.11 (dbc15_1_preview) (3)

HOW CAN I BE EVEN MORE EFFICIENT?

©2024 Databricks Inc. — All rights reserved

What if you never had to wait for the cluster to startup?

©2024 Databricks Inc. — All rights reserved

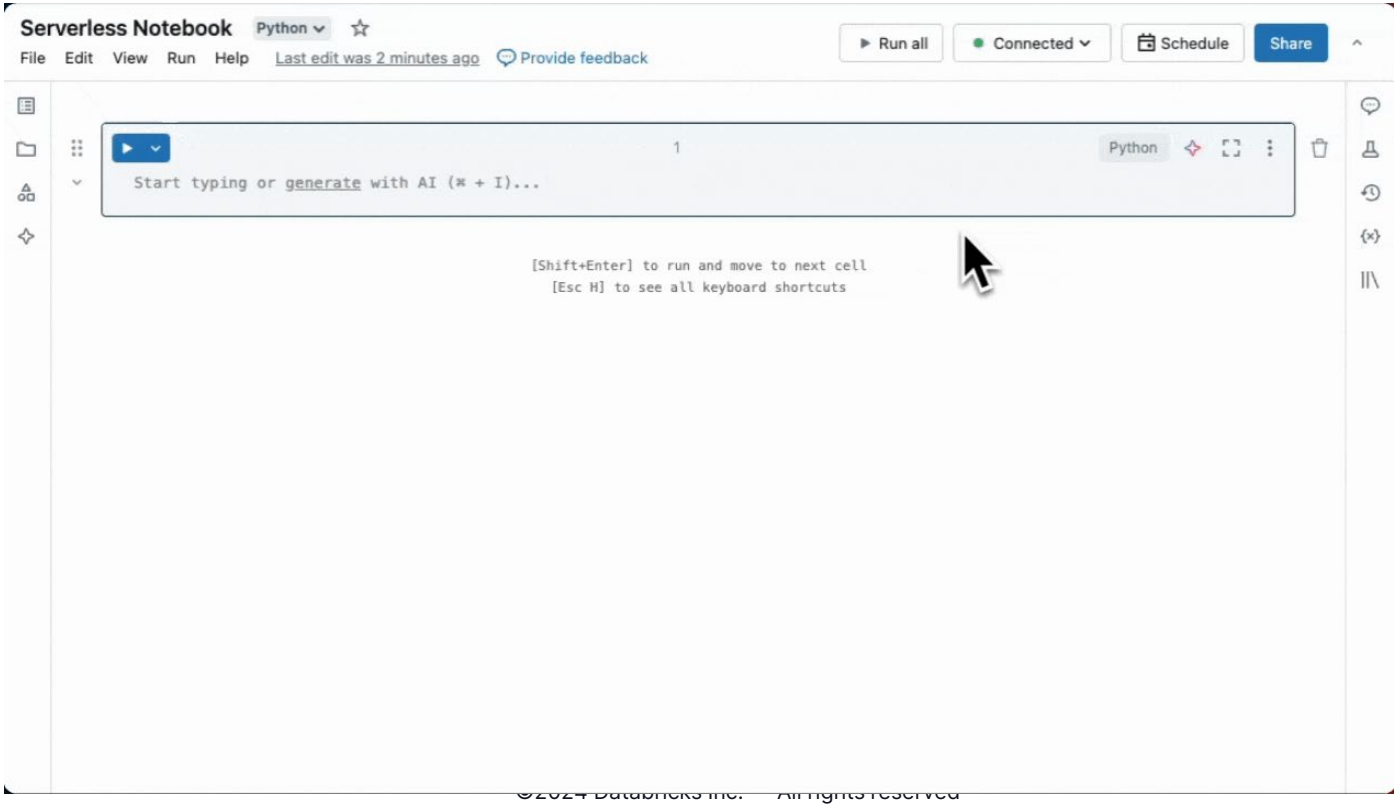
Serverless Compute

©2024 Databricks Inc. — All rights reserved

SERVERLESS NOTEBOOKS

Public Preview

So simple, it's hard to show!



©2024 Databricks Inc. All rights reserved

Imagine you connect from the IDE and it just runs….

©2024 Databricks Inc. — All rights reserved

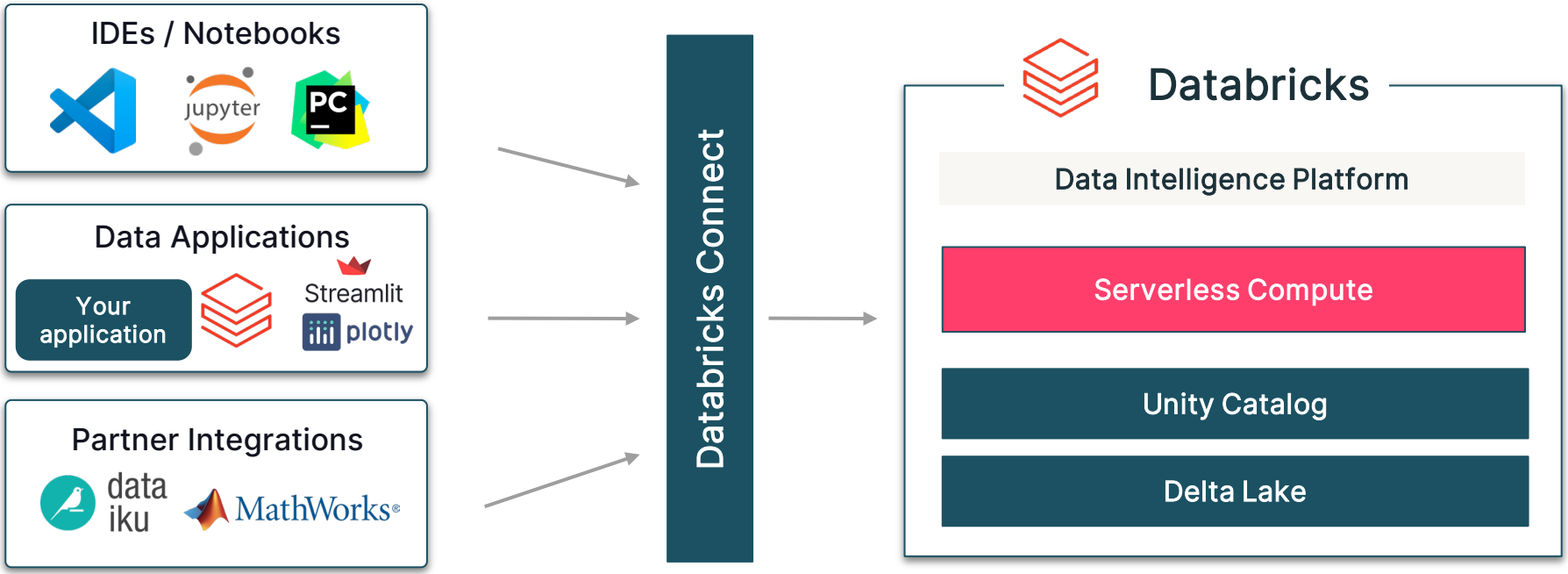
INTRODUCING

SERVERLESS DATABRICKS CONNECT

©2024 Databricks Inc. — All rights reserved

RUN ON SERVERLESS COMPUTE FROM ANYWHERE WITH DB CONNECT!

Public Preview (Soon)



©2024 Databricks Inc. — All rights reserved

SERVERLESS DB CONNECT

- Serverless enabled in your workspace
- Use your existing code
- Use your existing profiles
- Simply use “serverless()”

Local IDE

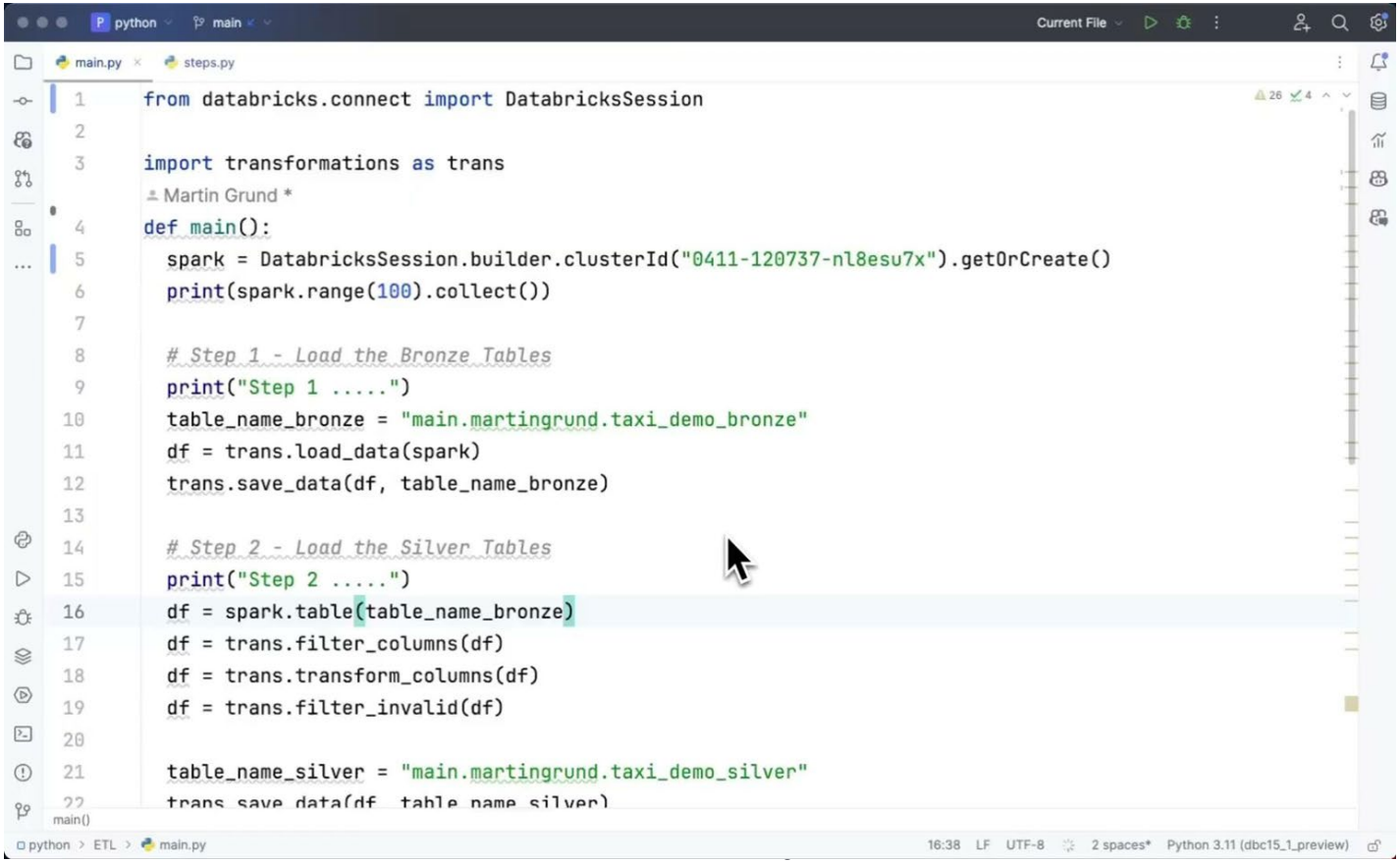

```
from databricks.connect import DatabricksSession

spark = DatabricksSession.builder.serverless().getOrCreate()


df = spark.read.table("samples.nyctaxi.trips")
df.count()
```


©2024 Databricks Inc. — All rights reserved

PYCHARM WITH SERVERLESS DB CONNECT



BENEFITS OF SERVERLESS DB CONNECT

Easy setup

- No need to create a cluster
- No cluster Id

Efficient everyday use

- Instant access with no idle time
- Dynamically adjust to workload and data size

Pay only for what you use

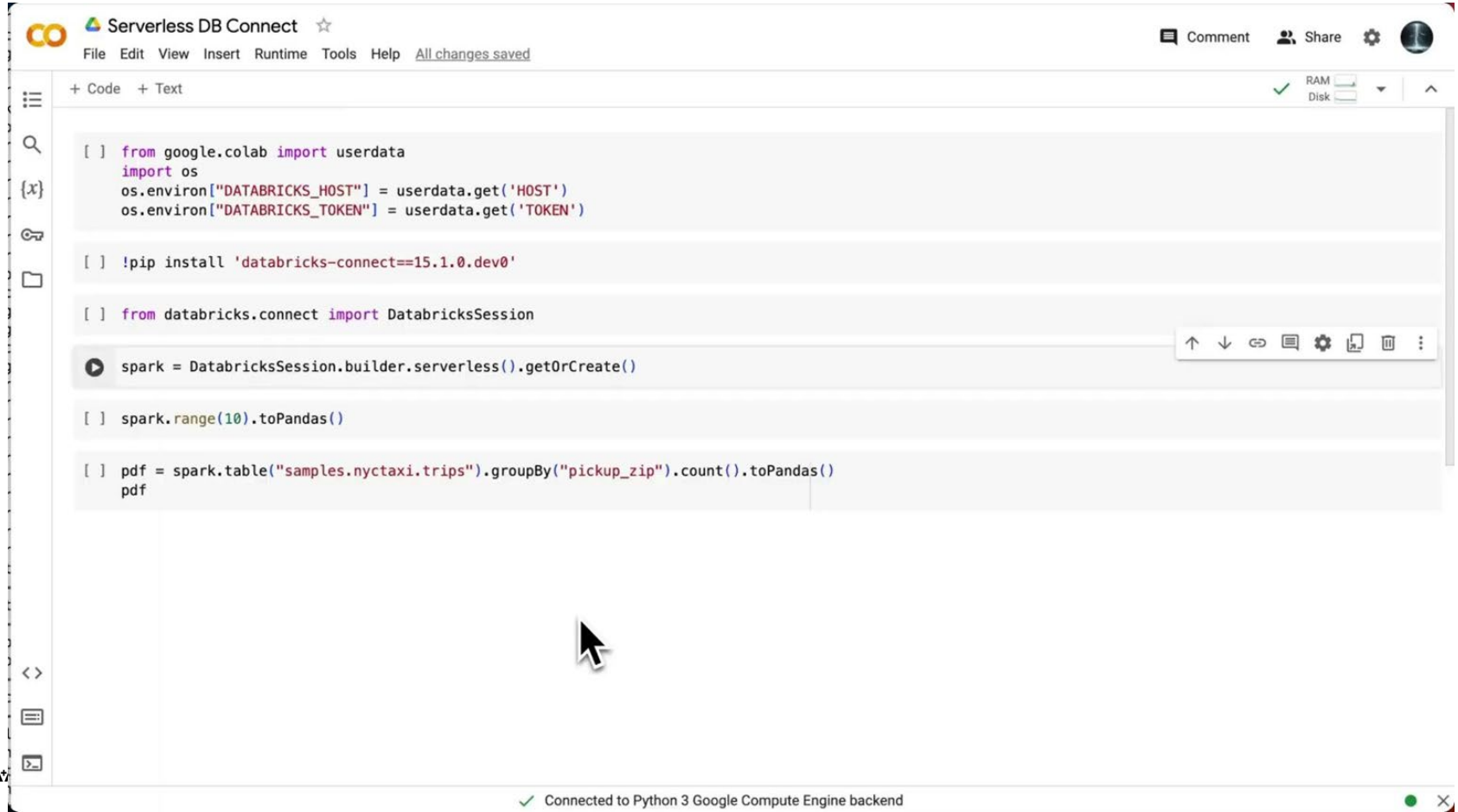
- Same cost as for

©2024 Databricks Inc. — All rights reserved

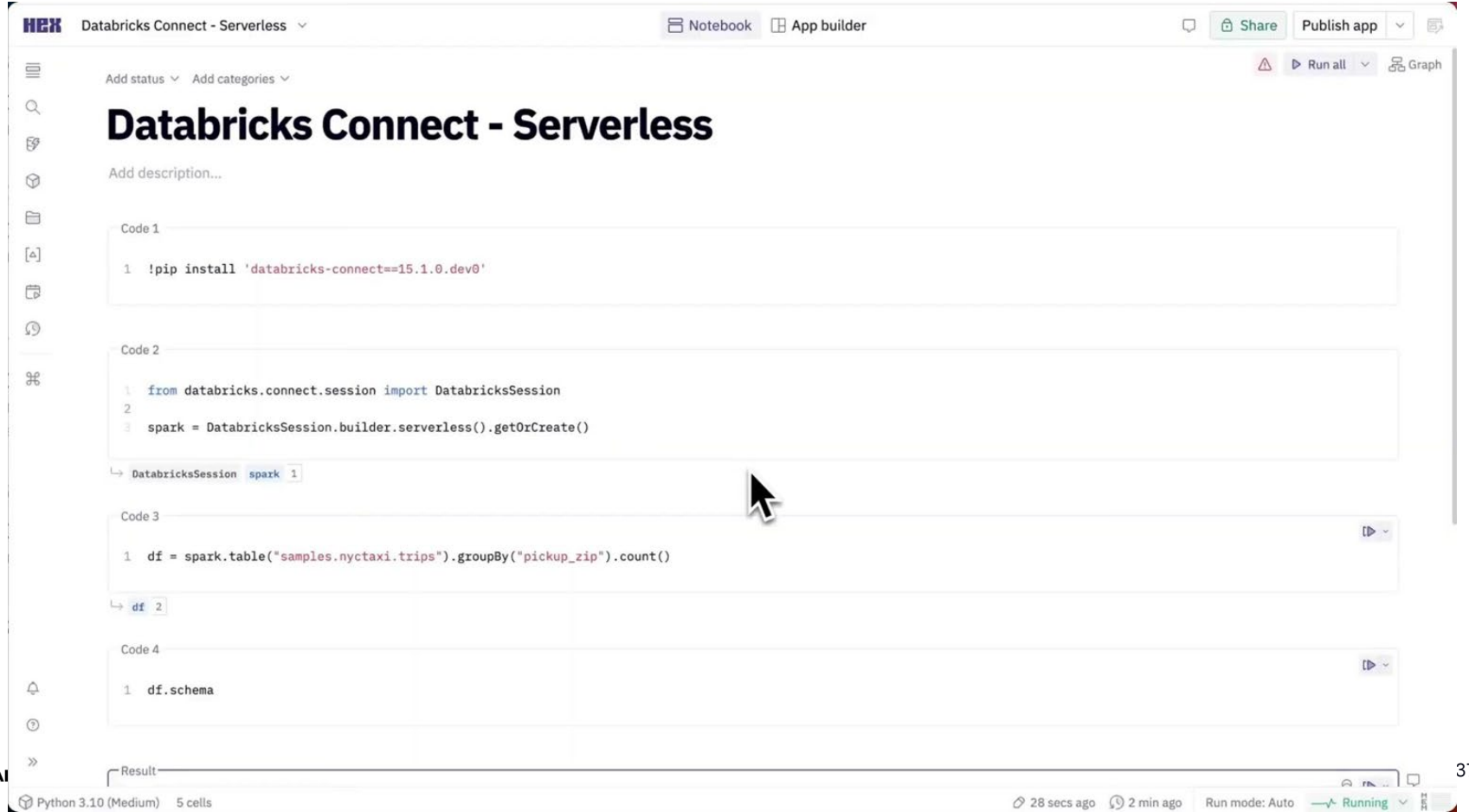
USE DB CONNECT FROM REALLY EVERYWHERE

©2024 Databricks Inc. — All rights reserved

Google Collab Databricks Connect



Hex ❤ Databricks Connect



SUMMARY

©2024 Databricks Inc. — All rights reserved

BUILD ON YOUR LAKEHOUSE FAST AND EFFICIENTLY

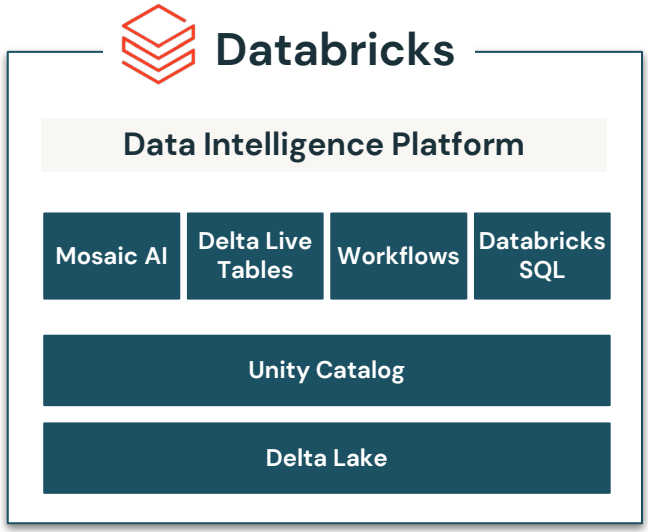
Databricks Connect + Serverless

DB Connect: Full power of Spark from anywhere

- Governance with Unity Catalog
- Authentication (OAuth, Azure AD, PAT)

Serverless: Fast and efficient

- No Cluster, No Cluster ID
- Immediate response & Scalability



©2024 Databricks Inc. — All rights reserved

SERVERLESS DATABRICKS CONNECT

Call to Action

Try DB Connect git examples:

- Develop ETL pipeline with DB Connect
- Build your first APP with DB Connect

Sign up to Serverless DB Connect Preview -> form in QR code

©2024 Databricks Inc. — All rights reserved

Questions?

©2024 Databricks Inc. — All rights reserved

Learn more at the summit!

Mobile App

Tells us what you think

- We kindly request your valuable feedback on this session.
- Please take a moment to rate and share your thoughts about it.
- You can conveniently provide your feedback and rating through the Mobile App

What to do next?

- Discover more related sessions in the mobile app!
- Visit the Demo Booth: Experience innovation firsthand!
- More Activities: Engage and connect further at the Databricks Zone!

Get trained and certified

Visit the Learning Hub Experience at Moscone West, 2nd Floor

- Take complimentary certification at the event; come by the Certified Lounge
- Visit our Databricks Learning website for more training, courses and workshops! databricks.com/learn

©2024 Databricks Inc. — All rights reserved