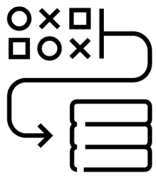
DATA+AI SUMMIT BY databricks

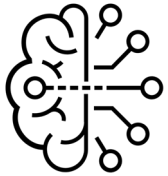
Dependency Management in Spark Connect: Simple, Isolated, Powerful

Akhil Gudesa, Hyukjin Kwon - R&D @ Databricks
June 2024

How can Dependencies Help You?

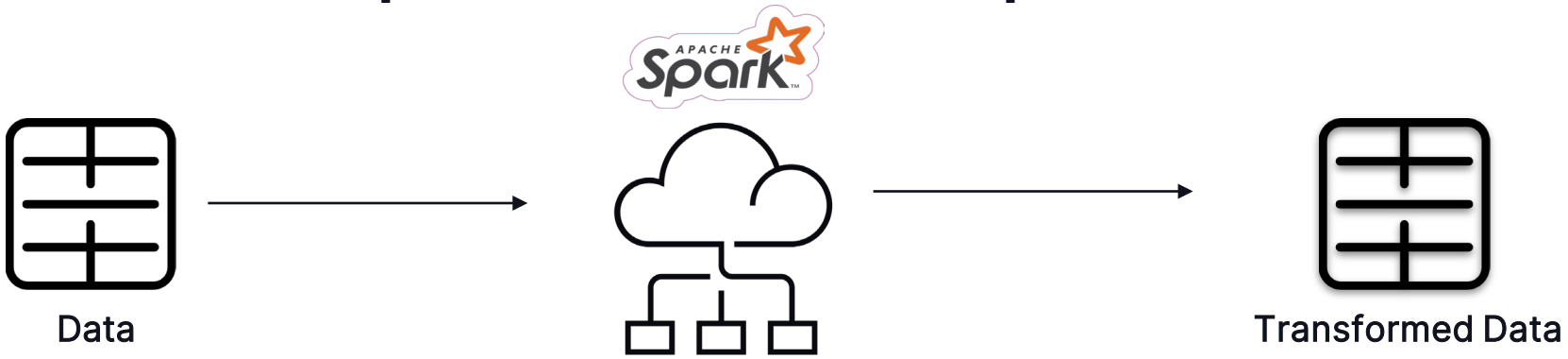


Custom ETL
Bring *your own, or 3rd party libraries* in your data transformation pipelines



Use OSS AI libraries
Have the freedom to experiment with the wide variety of existing ML open source libraries

How can Dependencies Help You?



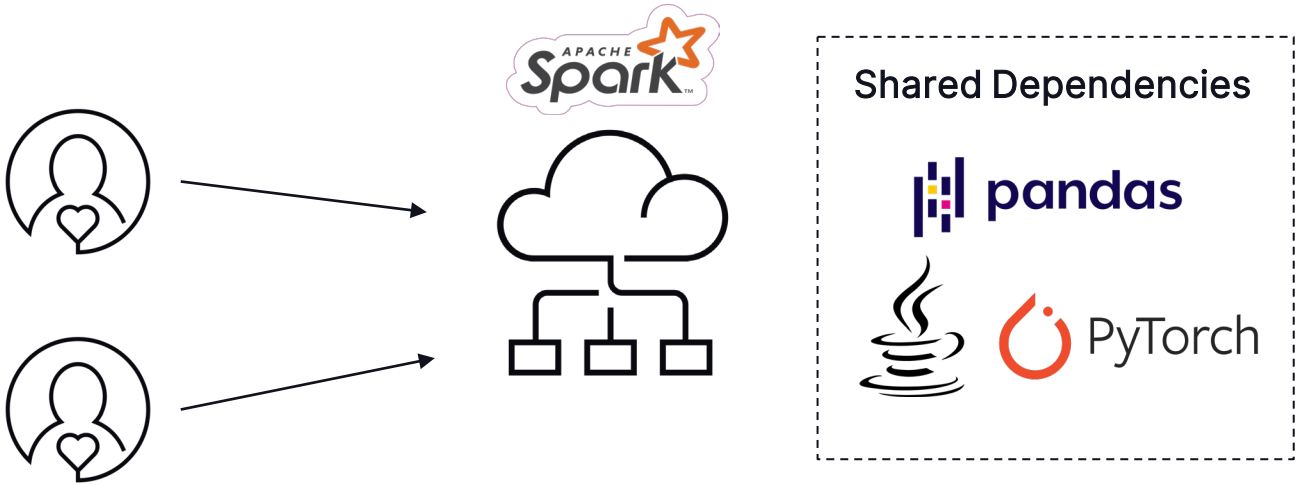
User Defined Function

Python Packages

JARs

Custom Libraries

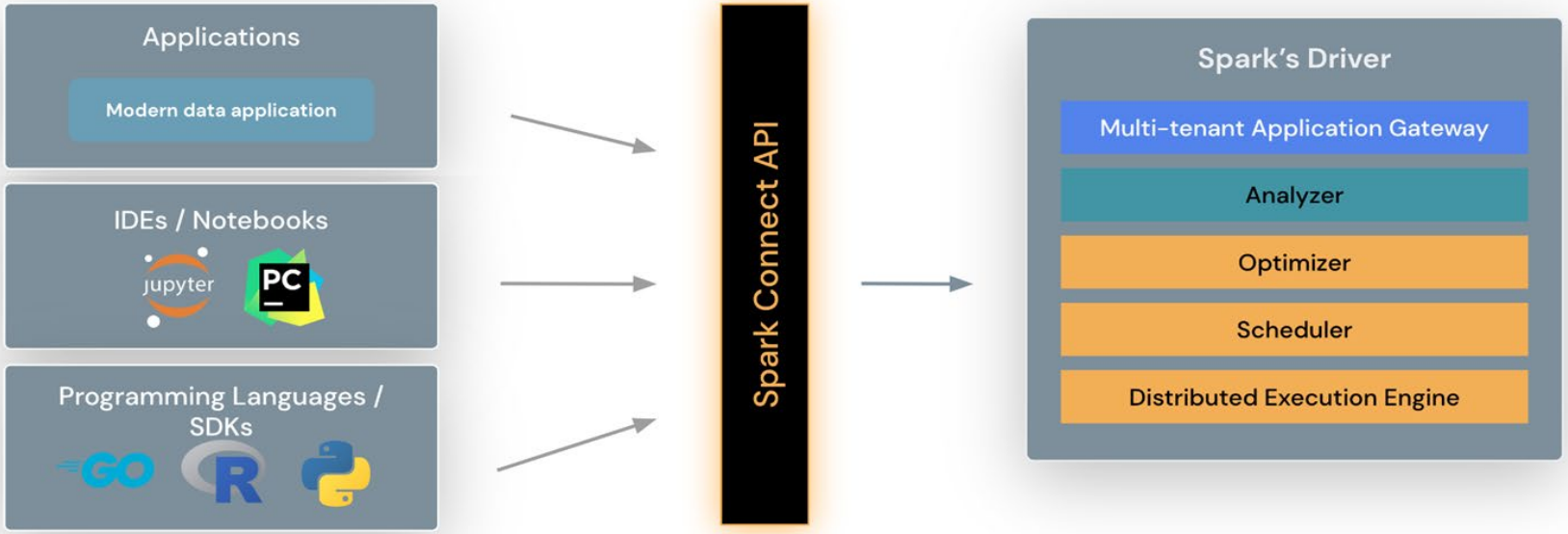
Dependencies in “Classic” Spark



- **Cluster Scoped**
  Environment/dependencies are shared between all users
- **Inter-User Conflicts**
  Users may require different versions of the same dependency
- **Static**
  Updating previously-set dependencies requires a Driver restart

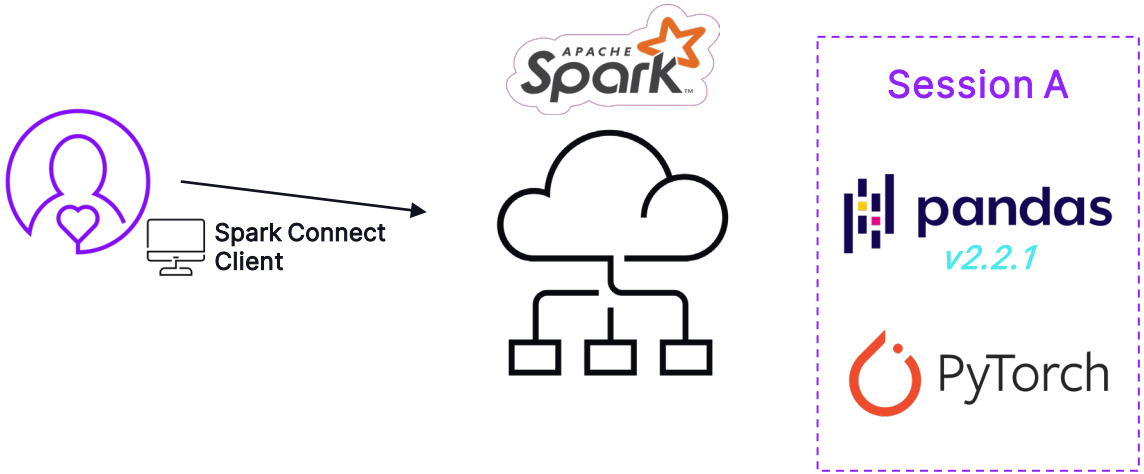
What is Spark Connect?

Thin client, with the full power of Apache Spark



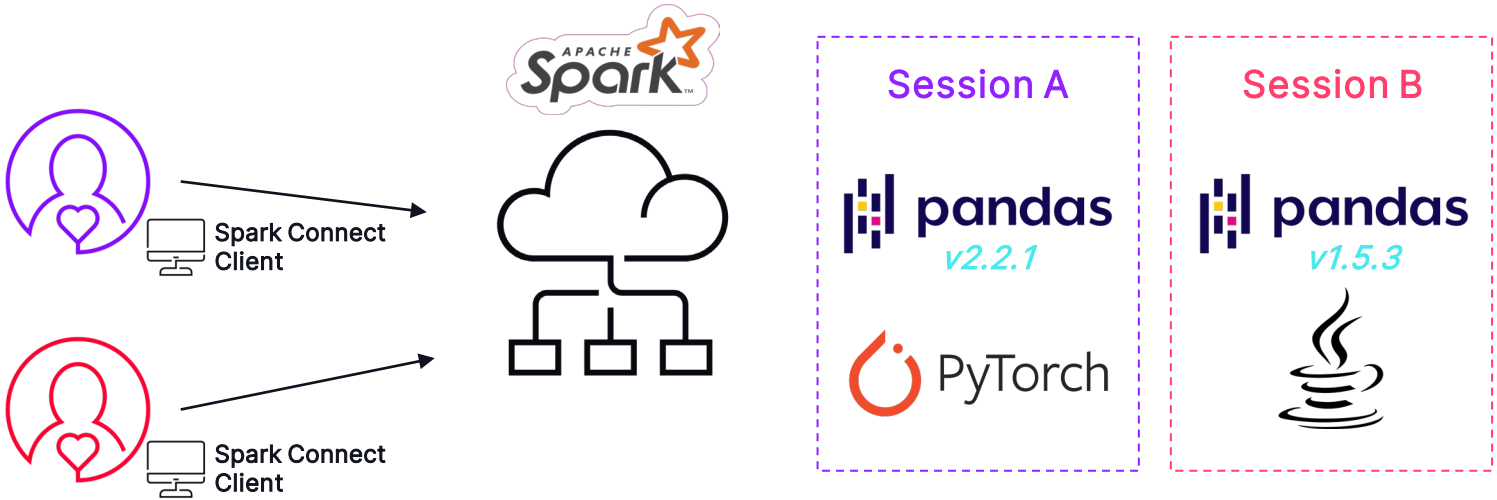
DEPENDENCY ISOLATION via Spark Connect

Dependencies in Spark Connect



Session Scoped
Environment/dependencies are scoped to a session

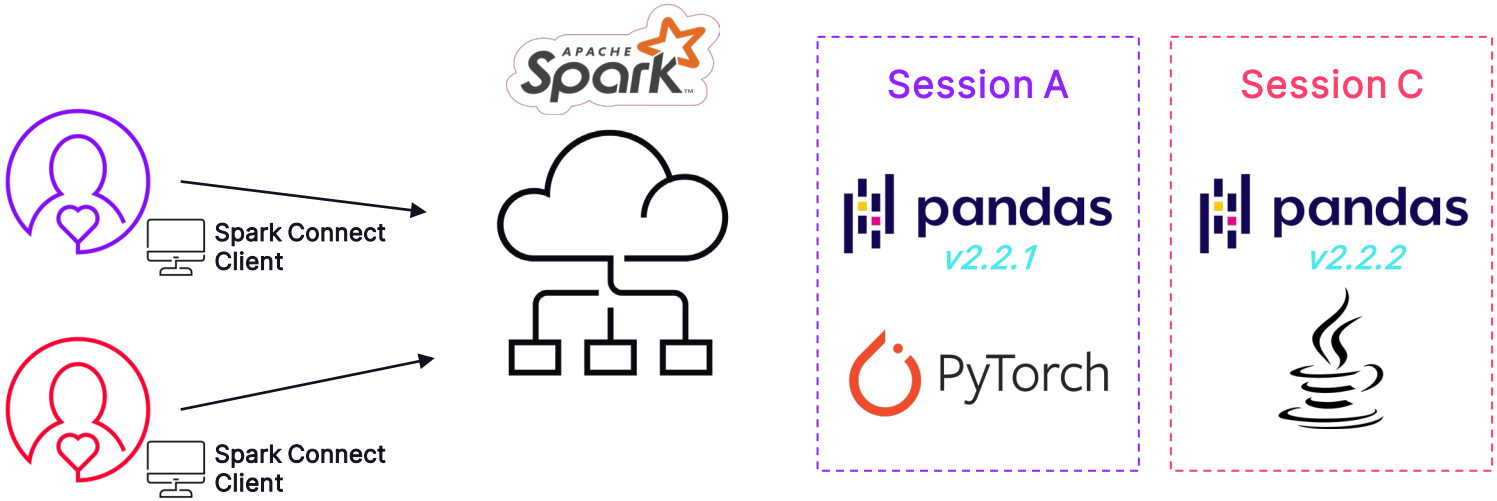
Dependencies in Spark Connect



Session Scoped
Environment/dependencies are scoped to a session

Inter-Session Isolation
Session resources do not interact with each other, free of inter-session dependency conflicts

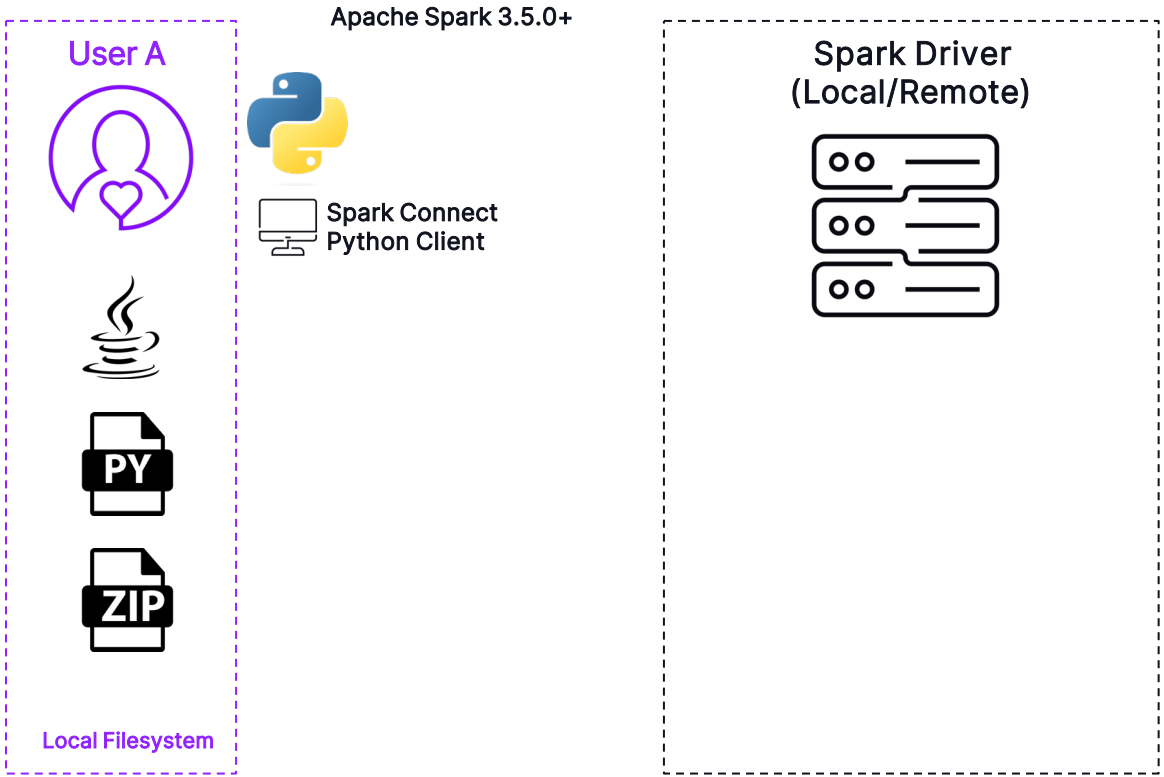
Dependencies in Spark Connect



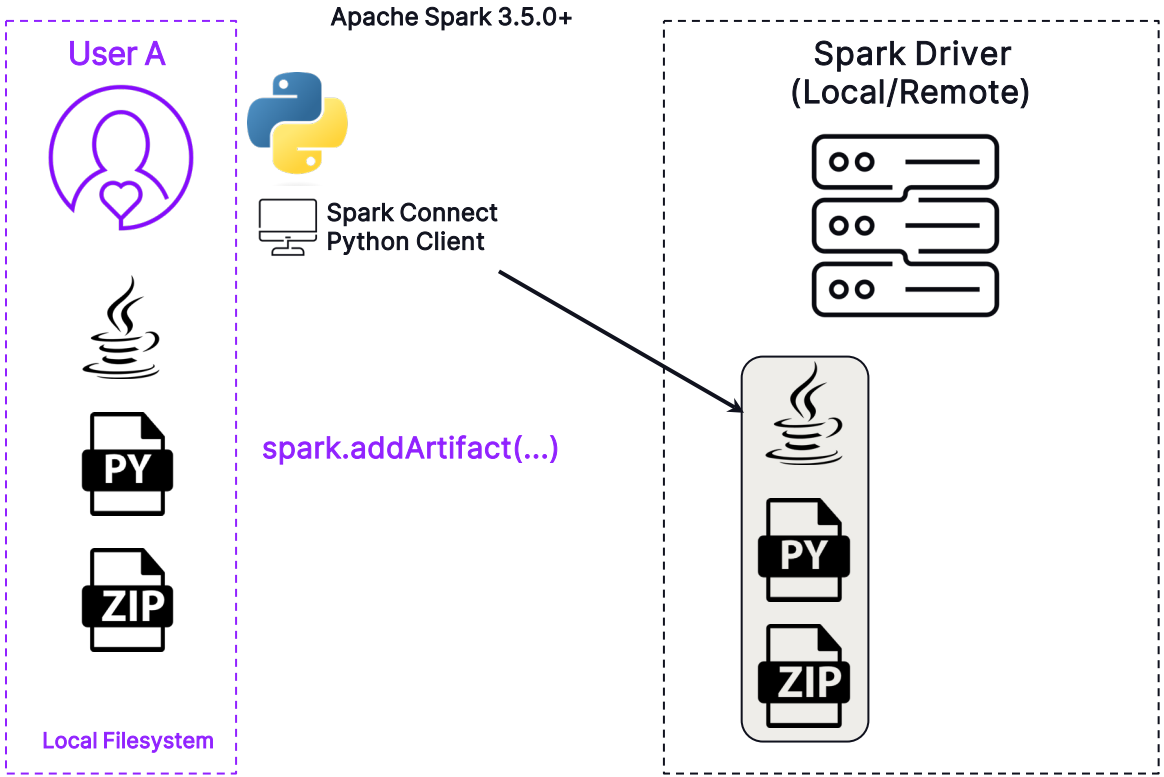
- **Session Scoped**
  Environment/dependencies are scoped to a session
- **Inter-Session Isolation**
  Session resources do not interact with each other, free of inter-session dependency conflicts
- **Dynamic**
  Use a new environment? No driver restart required; Simply specify environment in a new session

AddArtifact API

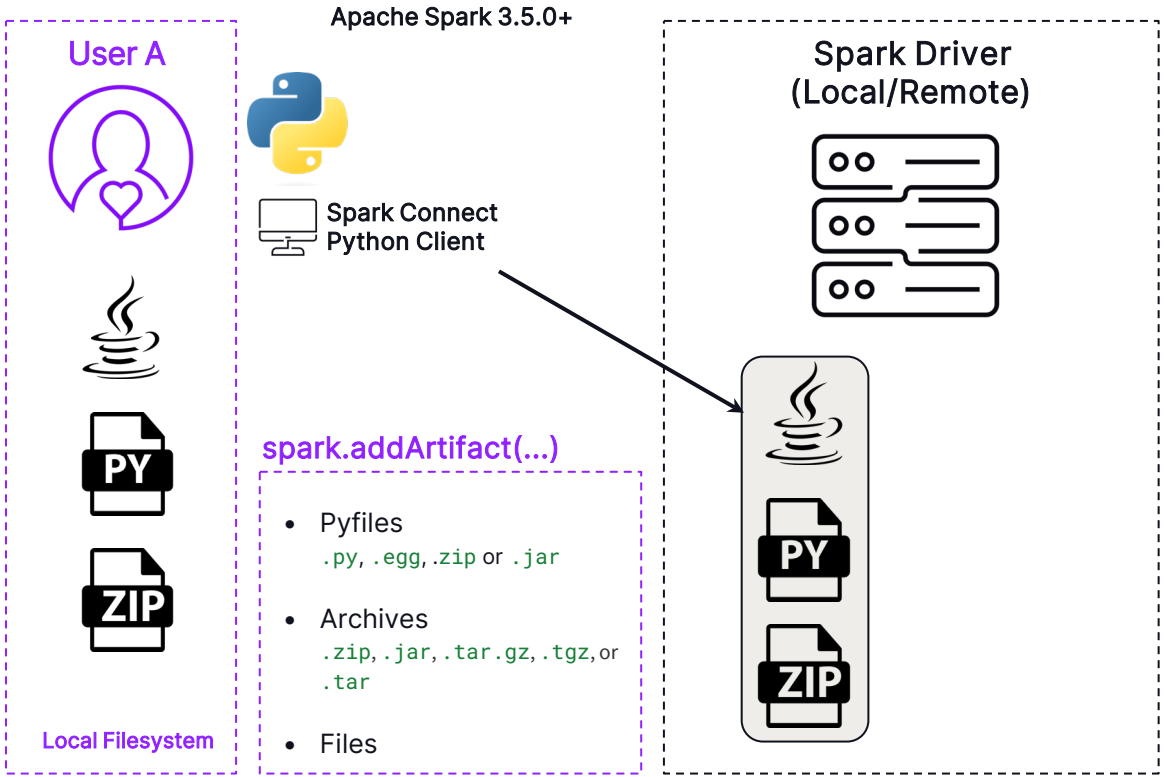
Leverage the AddArtifact API



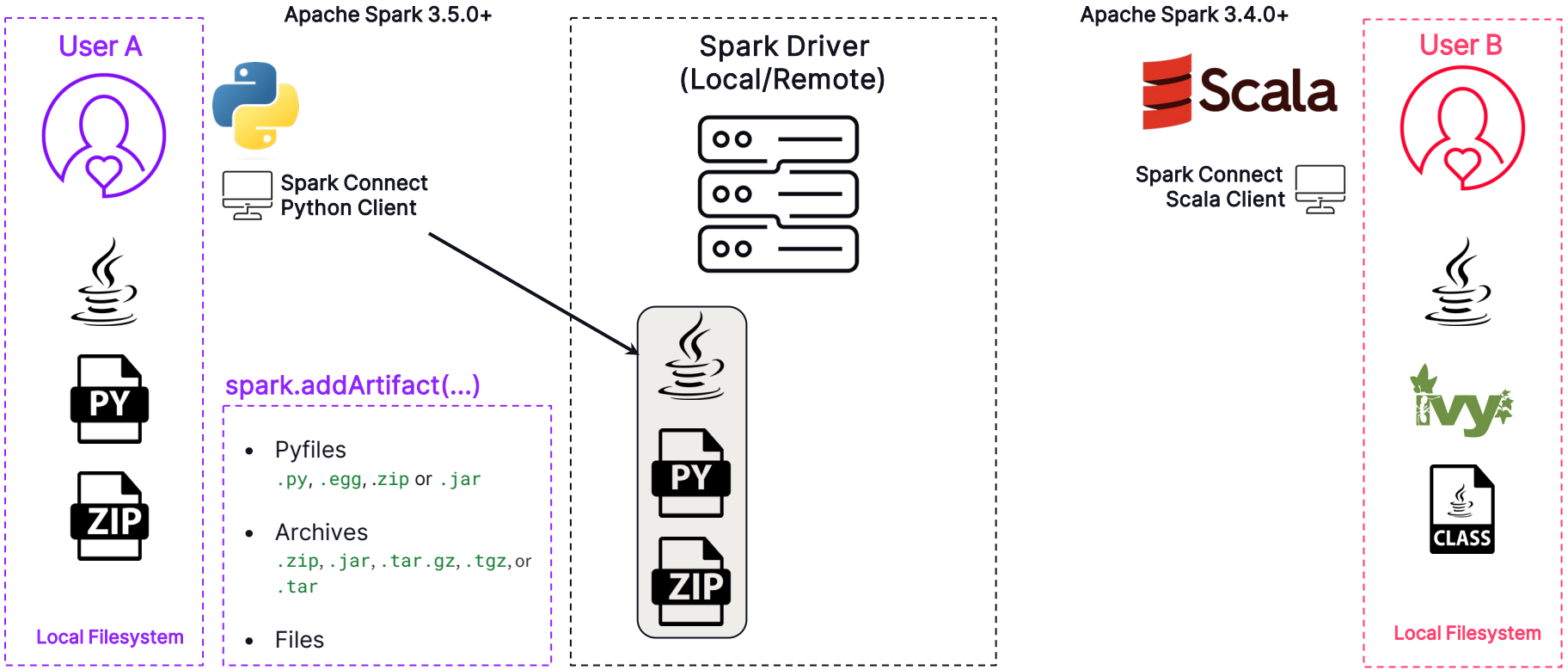
Leverage the AddArtifact API



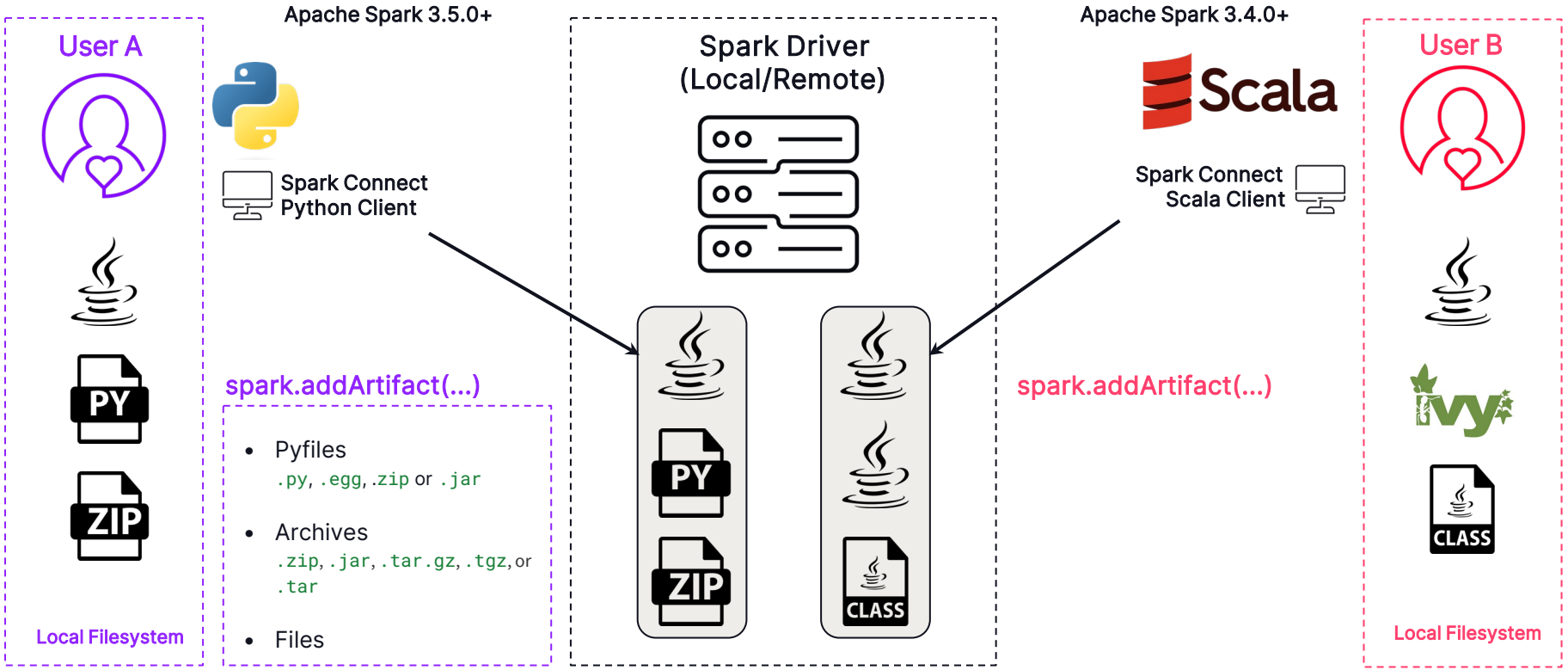
Leverage the AddArtifact API



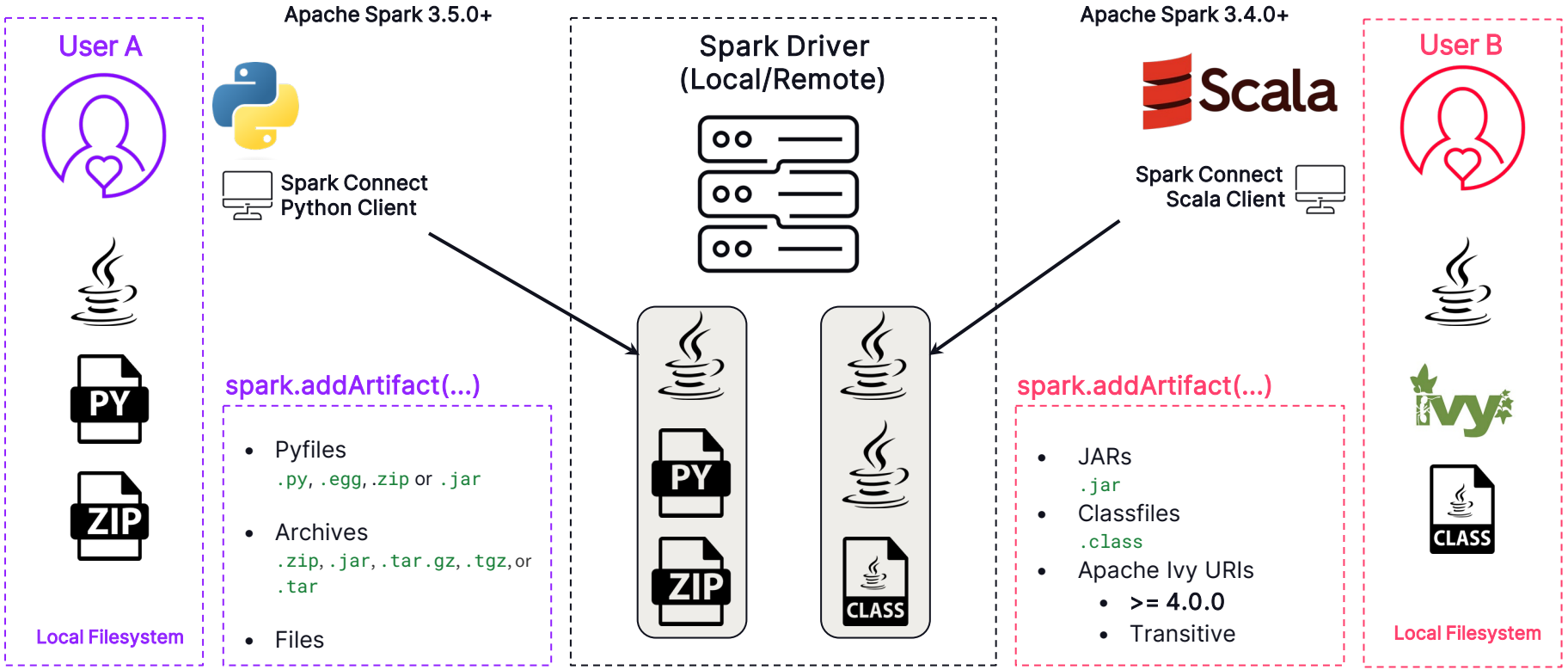
Leverage the AddArtifact API



Leverage the AddArtifact API



Leverage the AddArtifact API



Building apps in:

1/ PYTHON
2/ SCALA

1/

BUILDING APPS IN PYTHON

Building Applications in python

Code Template

Python

```python
from pyspark.sql import SparkSession

# 1. Setup Spark Connect session
spark = SparkSession.builder().remote("sc://localhost").build()

# 2. Upload files

# Arbitrary files
spark.addArtifact(<PATH_TO_PY_FILES>, file=True)

# .py, .egg, .zip or .jar, automatically added into PYTHONPATH
spark.addArtifact(<PATH_TO_PY_PYFILES>, pyfile=True)

# .zip, .jar, .tar.gz, .tgz, or .tar, automatically untarred in current working directory of UDF execution
spark.addArtifact(<PATH_TO_PY_ARCHIVES>, archive=True)
```

```
pip install pyspark-connect
```

Building Applications in python

Example Code

Python

```python
from pyspark.sql.functions import udf

spark = SparkSession.builder().remote(
    "sc://localhost").build()

spark.addArtifact("mylib.py", pyfile=True)

@udf("INT")
def my_udf(x):
   import mylib
   return mylib.my_calc(x)

my_df = spark.range(1)
my_df.select(my_udf("id").alias("output")).show()
```

mylib.py

```python
def my_calc(x):
   """
   My own calculation logic
   """
   return x * x
```

Building Applications in python

Example Code

Python

```
from pyspark.sql.functions import udf

spark = SparkSession.builder().remote(
    "sc://localhost").build()

spark.addArtifact("myarchive.tgz#mydir", archive=True)

@udf("ARRAY<STRING>")
def my_udf(x):
   import os
   return os.listdir("mydir")

my_df = spark.range(1)
my_df.select(my_udf("id").alias("output")).show()
```

myarchive.tar.gz

```
myfile1.txt
myfile2.txt
myfile3.txt
```

Building Applications in python

What's next?

Python

```
import conda_pack
import os

conda_pack.pack()
default_env = os.environ.get('CONDA_DEFAULT_ENV')

spark.addArtifact(
    f"{default_env}.tar.gz#environment", archive=True)

spark.conf.set(
    "spark.sql.execution.pyspark.python",
    "environment/bin/python")
```

mylib.py

```
@udf(packages=[
    "pandas==1.5.3", "pyarrow", "scikit-learn"])
def my_udf3():
    import pandas
    import pyarrow
    import scikit-learn
    # ...
```

UDF level control
(WIP)

2/

BUILDING APPS IN SCALA

Building Applications in Scala

Template

Scala

```scala
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.connect.client.REPLClassDirMonitor

// 1. Setup Spark Connect session
val spark = SparkSession.builder().remote("sc://localhost").build()

// 2. Register a ClassFinder to monitor and upload the classfiles from the build output.
val classFinder = new REPLClassDirMonitor(<ABSOLUTE_PATH_TO_BUILD_OUTPUT_DIR>)
spark.registerClassFinder(classFinder)

// 3. Upload JAR dependencies
spark.addArtifact(<ABSOLUTE_PATH_TO_JAR_DEP>)
spark.addArtifact(<APACHE_IVY_URI>)
```

Building Applications in Scala

Setup Build System


```
1  ThisBuild / version := "0.1.0-SNAPSHOT"
2  ThisBuild / scalaVersion := "2.13.14"
3
4  lazy val root = (project in file("."))
5    .settings(
6      name := "SparkConnectOSS4",
7      // Add library dependencies
8      libraryDependencies ++= Seq(
9        "org.apache.spark" %% "spark-sql-api" % "4.0.0-preview1",
10       "org.apache.spark" %% "spark-connect-client-jvm" % "4.0.0-preview1",
11       "net.moznion" % "random-string" % "1.1.0"
12     )
13   )
```

Building Applications in Scala

Sample Code

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.connect.client.REPLClassDirMonitor
import org.apache.spark.sql.functions.{col, udf}
import net.moznion.random.string.RandomStringGenerator

object Main {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().remote("sc://localhost").build()

    val sourceLocation = getClass.getProtectionDomain.getCodeSource.getLocation.getPath
    val classFinder = new REPLClassDirMonitor(sourceLocation)
    spark.registerClassFinder(classFinder)

    val ivyString = "ivy://net.moznion:random-string:1.1.0"
    spark.addArtifact(ivyString)

    def random(x: Int): String = s"${x}_" + new RandomStringGenerator().generateFromPattern("ccCC!!")
    def random_udf = udf(random _)

    spark.range(3)
      .withColumn("random", random_udf(col("id")))
      .select("random")
      .show()
  }
}
```

Setup Session

Setup Classfile upload

Upload dependencies

Use dependencies (UDF)

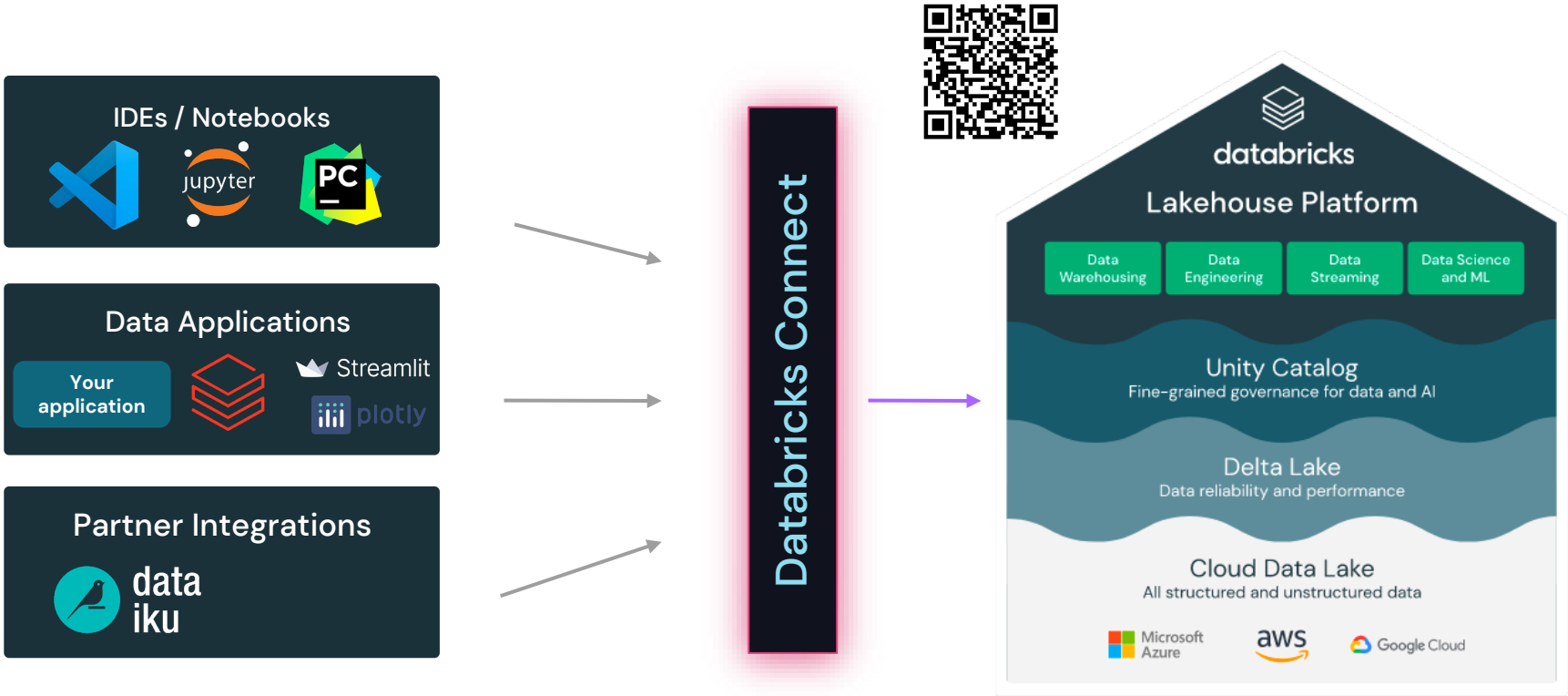
Execute UDF

```
+--------+
|  random|
+--------+
|0_pwQR%]|
|1_wlAT#<|
|2_uaNN^+|
+--------+
```

DEMO

ON DATABRICKS?

Connect to Databricks from anywhere!



Databricks ❤️ OSS

Python

Spark Connect

```
from pyspark.sql import SparkSession

spark = SparkSession.builder().remote("sc://localhost").build()
```

Databricks Connect

```
from databricks.connect import DatabricksSession

spark = DatabricksSession.builder.profile("<profile-name>").getOrCreate()
```

Databricks ❤️ OSS

Scala

Spark Connect

```scala
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder().remote("sc://localhost").build()

val sourceLocation = getClass.getProtectionDomain.getCodeSource.getLocation.getPath
val classFinder = new REPLClassDirMonitor(sourceLocation)
spark.registerClassFinder(classFinder)
```

Databricks Connect

```scala
import com.databricks.connect.DatabricksSession
import com.databricks.sdk.core.DatabricksConfig

val sourceLocation = getClass.getProtectionDomain.getCodeSource.getLocation.toURI

val config = new DatabricksConfig().setProfile("<profile-name>")
val spark = DatabricksSession.builder().sdkConfig(config).addCompiledArtifacts(sourceLocation).getOrCreate()
```

Call to Action

Spark Connect docs

Databricks Connect docs

DAIS sessions on Spark/Databricks Connect

"Ten-Times Easier Development Using Databricks Connect and Serverless Spark"
Intermediate, 13th June 12:30pm

"An In Depth Look at the New Features of Apache Spark 3.5"
Intermediate, 13th June 4:00pm

"What's Next for the Upcoming Apache Spark 4.0?"
Intermediate, 12th June 11:20am

DATA+AI SUMMIT