DATA+AI SUMMIT 2022

Cutting the edge in fighting cybercrime: reverse-engineering a search language to cross-compile it to PySpark

Jude Ken-Kwofie
Principal Engineer, HSBC Cybersecurity Sciences & Analytics

Abigail Shriver
Lead Cybersecurity Engineer, HSBC Cybersecurity Sciences & Analytics

Serge Smertin
Senior Resident Solutions Architect, Databricks

ORGANIZED BY databricks

databricks

HSBC

HSBC is a multinational investment and financial services company.

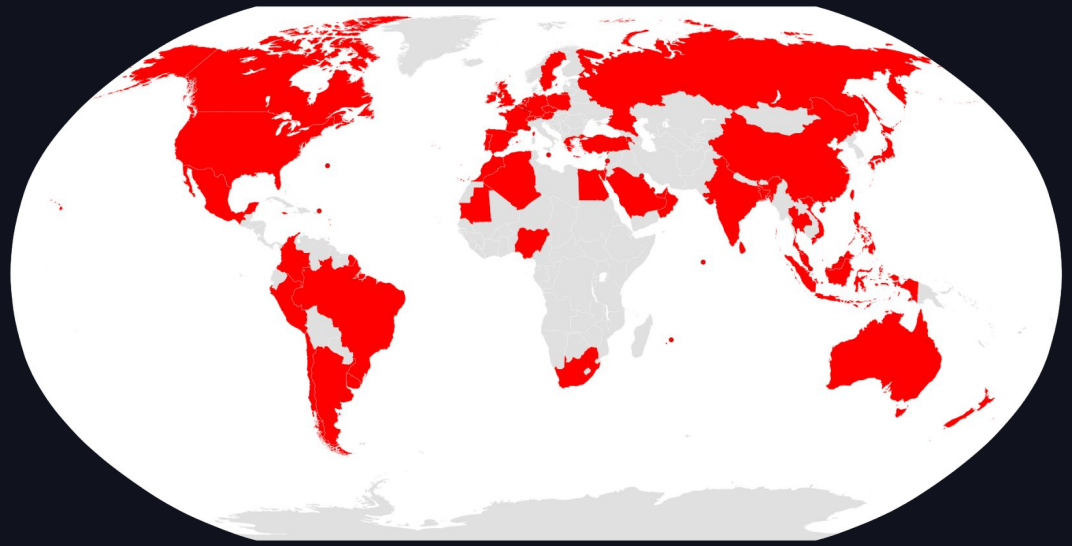
226k+ Employees

40m Customers

~64 Countries

~1.3m Assets

4 Main Cloud Providers



$4.24m
Average Breach Cost

$401m
Mega Breach Cost

The scale of cybersecurity at HSBC

1,880+ Global cybersecurity team members

16 countries and territories with dedicated cybersecurity centers

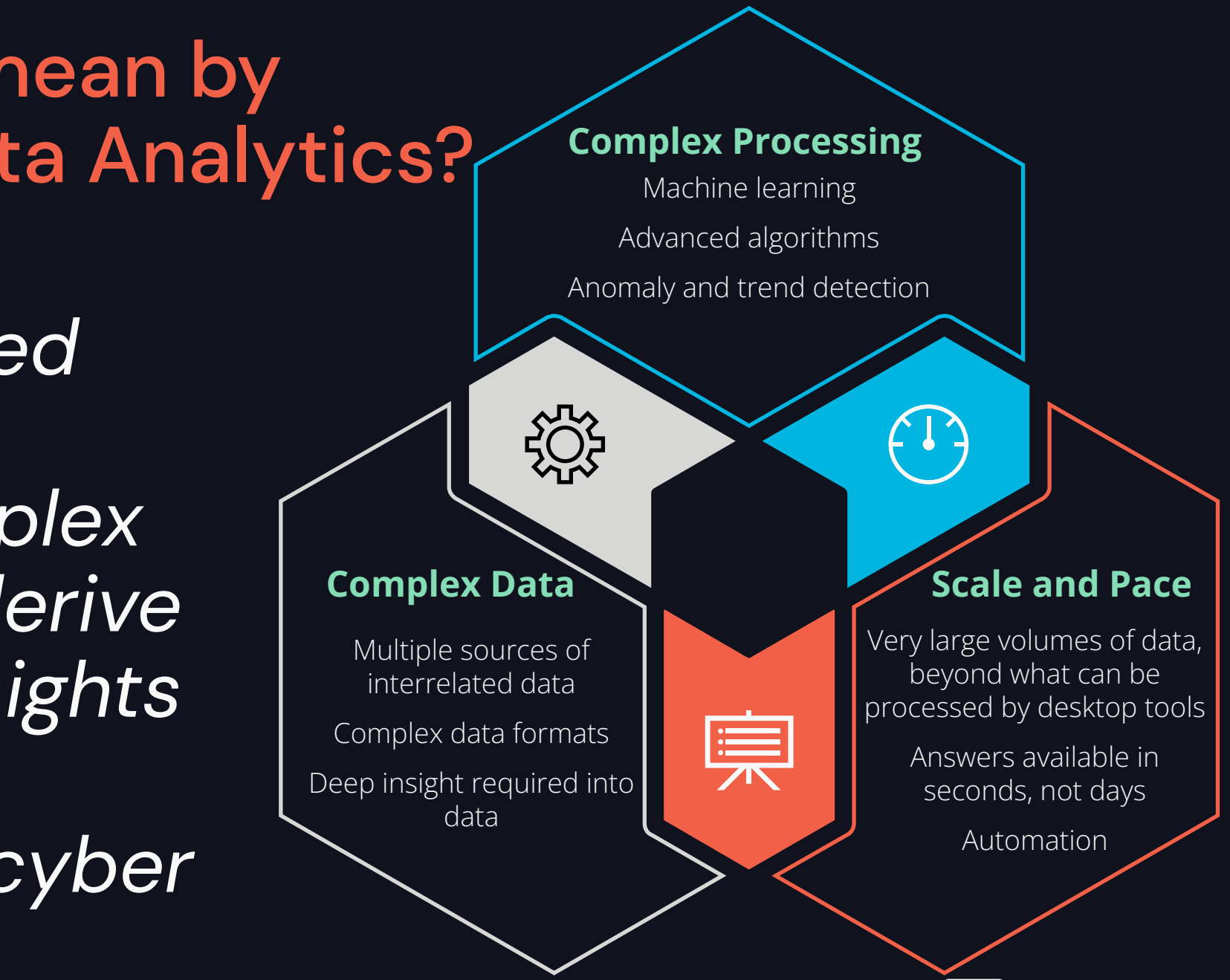
200+ Global organisations receive intelligence feeds from HSBC

75+ Global cybersecurity education events held for employees each year

What do we mean by Advanced Data Analytics?

Using advanced algorithms on massive, complex data sets to derive actionable insights at pace …

… for cyber



Introductions

Jude Ken-Kwofie

Principal Software Engineer
Cybersecurity Sciences & Analytics
HSBC

Jude Ken-Kwofie is a Principal Software Engineer at HSBC. He has over 15 years of experience across engineering, data systems, and security. He is the author of several telecom battery standards and led projects for large-scale financial services and adtech data solutions. At HSBC, Jude's role is to help engineer next-gen Cybersec analytic capabilities. When not working, Jude is busy with his Business PhD researching alternative small business financing and decision support systems.

Abigail Shriver

Lead Cybersecurity Engineer
Cybersecurity Sciences & Analytics
HSBC

Abigail Shriver, Lead Cyber Security Engineer at HSBC's Cybersecurity Sciences and Analytics division, develops in-house data-driven endpoint, cloud, & network security analytics. Previously, she developed the largest centralized data lake within Capital One and leveraged the PBs of data within the lake to identify anomalies & remediate vulnerabilities. She attended The George Washington University studying CS and IA concentrating in Cyber Security & Security Policy.

Serge Smertin

Senior Resident Solutions Architect
Databricks

In his over 15 years of career, Serge has been dealing with data solutions, cybersecurity, and heterogeneous system integration. His track record got novel ideas from whiteboard to operating them in production for years, like large-scale malware forensic analysis for the cyber-threat intelligence, or real-time data science platform as the basis for anomaly detection and decision support systems for an industry-leading payments service provider.

DATA+AI SUMMIT 2022

1. Intro
2. Traditional SIEMs for Cyber
3. Migration in Practice
4. Business Outcomes
5. Search Processing Language Transpiler
6. Questions

HSBC

DATA+AI SUMMIT 2022

HSBC

Traditional SIEMs for Cyber

Jude Ken-Kwofie
Principal Engineer
HSBC Cybersecurity Sciences & Analytics

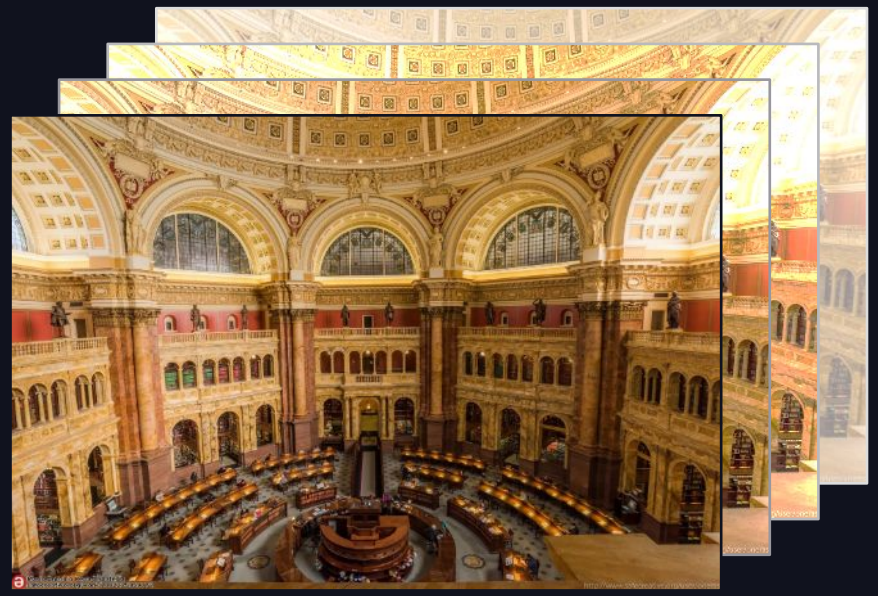
ORGANIZED BY databricks

Cybersecurity is a massive big data problem

100-200 TB/day

x 13 months = 38-79 PB

Everything is an asset



Storing 10x the contents of every book in the US Library of Congress, every day

Traditional Security Information and Event Management (SIEM)

Cyber Analysts prefer SIEMs and Search Processing Languages

SIEM Cybersec Analysis

Cybersec Analysts / Threat Hunter

```
index=*dns*
|sourcetype=*cloud*
|stats sum(bytes_out)
|rename sum(bytes_out) AS "Total Bytes"
|table domain_name ….
```

SIEM

search and filter → munge → report → cleanup

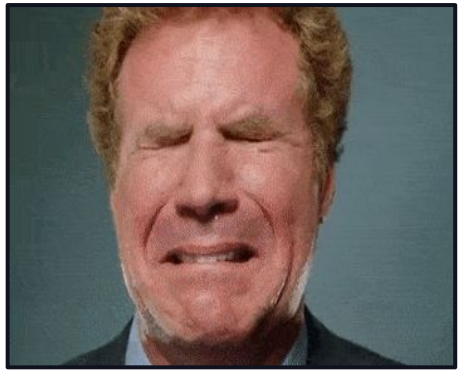
Traditional SIEMs Engineered for Processing Recent Data

Traditional SIEM Processing > 100TB+ / day is Challenging

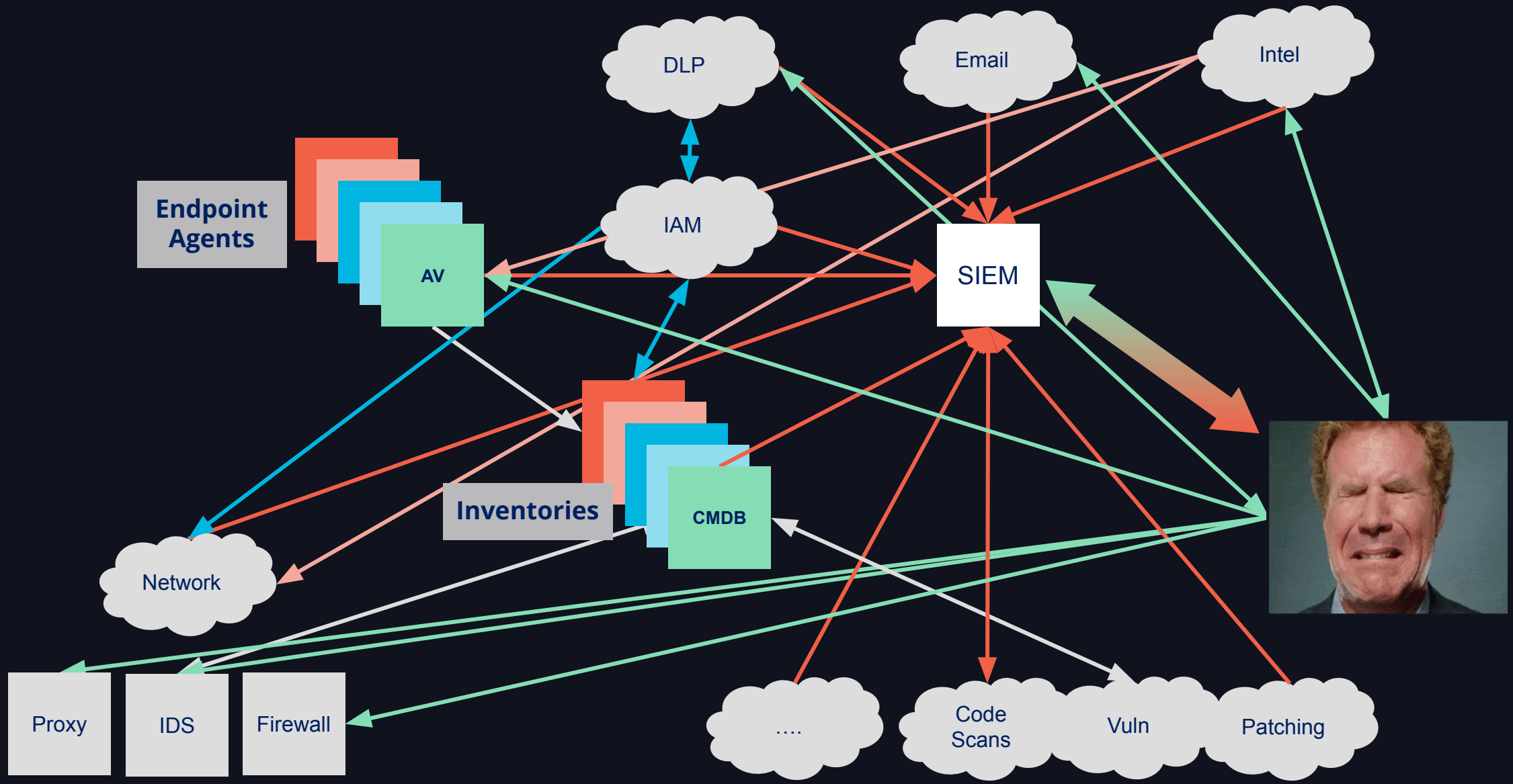
- SIEMs struggle with processing large data volumes, **>30 TB/day**
- Cybersecurity Data Volumes > **100–200 TB / day**
- **$$$$**
- **Joins and Aggregations**

- **HSBC Cyber Lakehouse** SIEM developed to:
  - Overcome challenges
  - Empower Cyber analysts and Threat Hunters

Lift-n-Shift is Challenging



So much data – so hard to get insight



Lift-n-Shift Threat Detections to PySpark

Benefits of Approach

- Process > 100TB / Day
- Empower SIEM economically
- Use advanced analytics & ML
- Near real-time detections
- Extensible

Search & Processing Language Analytic

```
index=*dns*
|sourcetype=*cloud*
|stats sum(bytes_out)
|rename sum(bytes_out) AS “Total Bytes”
|table domain_name ….
```

Migration via Transpiler

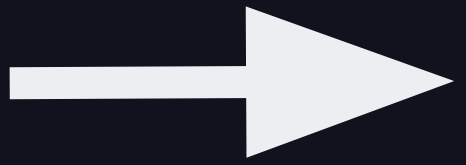
Transpiling Search Processing Language Code

Example

```scala
import spl.Transpiler

println(Transpiler.toPython("""
index=main
(sourcetype::ProcessRollup2* OR sourcetype::CommandHistory*)
(TERM(ProcessRollup2) OR TERM(CommandHistory))
TERM(TODO_BACKSLASHES)
   (
   CommandHistory="*\\XXX\\XXXX\\XXXX\\XXXX\\XXX*"
   OR
   CommandLine="*\\XXX\\XXXX\\XXXX\\XXXX\\XXX*"
   )
   earliest=-1h

|| rex max_match=0 field=_raw "(?i)(?<IOC>(YYYYYY|YYYYYY|YYYYY))"
|| stats count earliest(_time) as earliest latest(_time) as latest
   values(ComputerName) as ComputerName by aid
|| eval x = strftime(earliest, "askjdhlakshdjashd")
|| table aid
         ComputerName
         LocalAddressIP4
         FileName
         event_simpleName
         IOC
         count
         CommandLine
         earliest
         latest
         earliest_h
         latest_h
"""))
```



DATA+AI SUMMIT 2022

Migration in practice

How it went

Abigail Shriver
Lead Cybersecurity Engineer
HSBC Cybersecurity Sciences & Analytics

ORGANIZED BY databricks

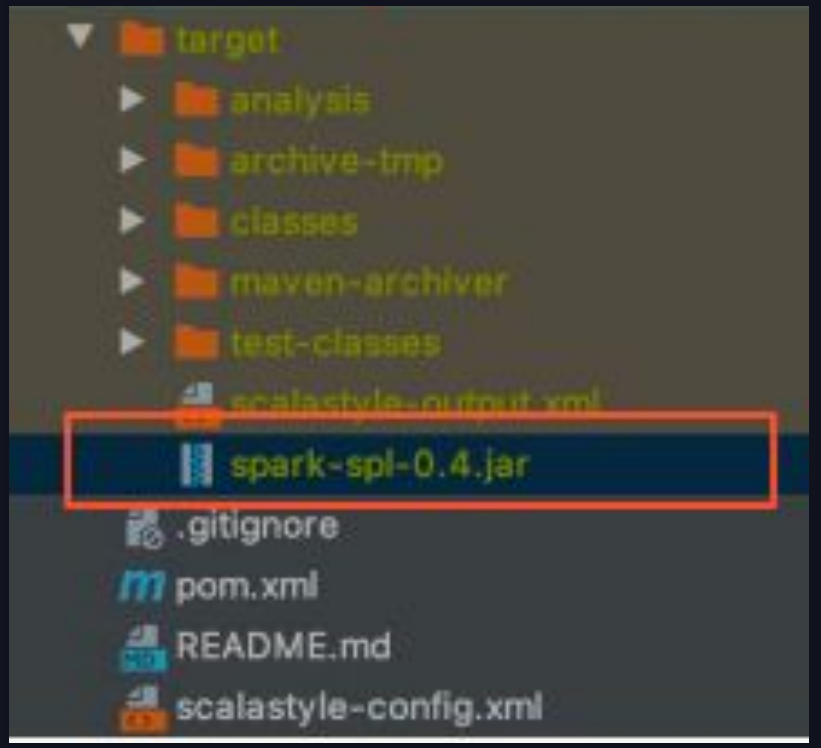
HSBC

Support Several Functions/Commands

Commands		Functions		Extensions
search	head	strftime()	memk()	term()
eval	format	values()	rmunit()	
table	fillnull	latest()	rmcomma()	
rex	eventstats	earliest()	ctime()	
regex	dedup	if()	auto()	
where	makeresults	mvcount()	num()	
lookup	mvexpand	coalesce()	replace()	
bin	streamstats	mvindex()	lower()	
rename	addtotals	mvappend()	none()	
join	multisearch	null()	CIDR search	
fields		min()		
convert		round()		
collec		max()		
stats		substr()		
sort		isnotnull()		
return		sum()		
mvcombine		mvfilter()		
map		len()		
inputlookup		count()		

How to Build the Transpiler?

- Built using **maven (3.6.x)**
  - **mvn -DskipTests - Plocal package**
- A (fat) **jar** will be created containing the code & all the dependencies required
- Created by default in **target** folder in the root directory of the repository



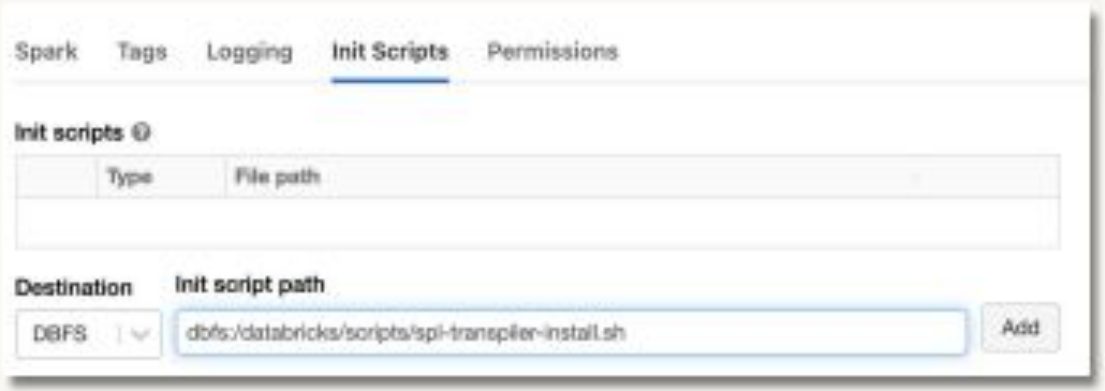
Attach to a Databricks Cluster

- Attach the created jar through the **Libraries** tab in the cluster configuration page in UI.

- *(Or)* Use Databricks Cluster **init-script** *(required for extensions like TERM())*

```
dbutils.fs.put("/databricks/scripts/spl-transpiler-install.sh",
"""
#!/bin/bash

cp /dbfs/tmp/spl/spark_spl_0_3.jar /databricks/jars/spark_spl_0_3.jar
""", True)
```



Cybersecurity Lakehouse Overview

Endpoint Custom Detections

- **100s** of queries to migrate over to PySpark on Databricks

- Endpoint detection data is already ingested into the cybersecurity lakehouse and registered in the metastore

Search Processing Language to PySpark

```
index=main
(sourcetype::ProcessRollup2*)
(TERM(ProcessRollup2))
TERM(TODO_BACKSLASHES)
    (
      CommandLine="*\\XXX\\XXXX\\XXXX\\XXXX\\XXX*"
    )
    earliest=-1h
| rex max_match=0 field=_raw "(?i)(?<IOC>(YYYYYY|YYYYYY|YYYYY))"
| stats count earliest(_time) as earliest latest(_time) as latest
    values(ComputerName) as ComputerName by aid
| eval x = strftime(earliest, "askjdhlakshdjashd")
| table aid
        ComputerName
        LocalAddressIP4
        FileName
        event_simpleName
        IOC
        count
        CommandLine
        earliest
        latest
        earliest_h
        latest_h
```

TRANSPILER

```
(spark
    .table('main')
    .where(F.expr("sourcetype LIKE 'ProcessRollup2%'"))
    .where(F.expr("term('ProcessRollup2')"))
    .where(F.expr("term('TODO_BACKSLASHES')"))
    .where(F.expr("CommandLine LIKE '%\\\\XXX\\\\XXXX\\\\XXXX\\\\XXXX\\\\XXX%'"))
    .where(F.col('_time') >= (F.expr('now()') + F.expr("INTERVAL '-1 hours'"))))
    .withColumn('IOC', F.regexp_extract(
      F.col('_raw'),
      '(?i)(?<IOC>(YYYYYY|YYYYYY|YYYYY))', 1)
    )
```

Enrichment Logic

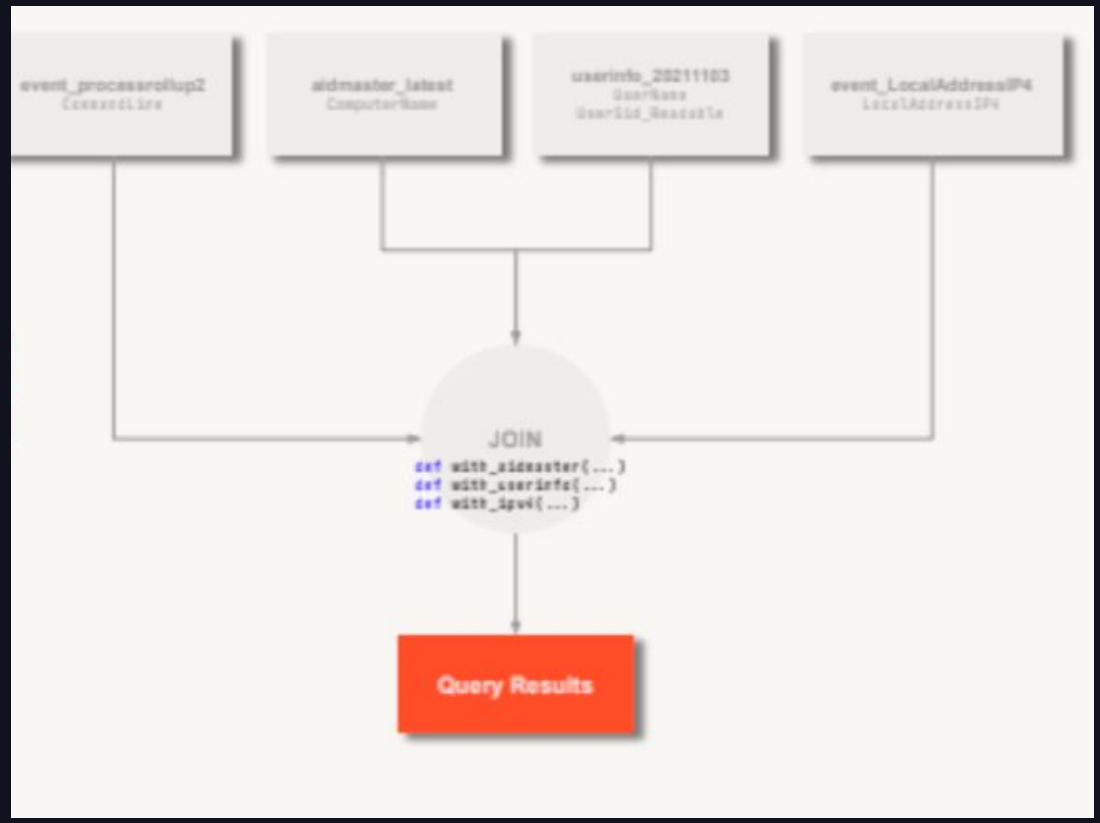
```
    .orderBy(F.col('_time').asc())
    .groupBy('aid')
    .agg(
      F.count(F.lit(1)).alias('count'),
      F.first(F.col('_time'), True).alias('earliest'),
      F.last(F.col('_time'), True).alias('latest'),
      F.collect_set(F.col('ComputerName')).alias('ComputerName'))
    .withColumn('x', F.date_format(F.col('earliest'), 'askjdhlakshdjashd'))
    .select(
      F.col('aid'),
      F.col('ComputerName'),
      F.col('LocalAddressIP4'),
      F.col('FileName'),
      F.col('event_simpleName'),
      F.col('IOC'),
      F.col('count'),
      F.col('CommandLine'),
      F.col('earliest'),
      F.col('latest'),
      F.col('earliest_h'),
      F.col('latest_h')))
```

Data in Search Processing Platform != Format in Databricks

Data in Search Processing Platform != Format in Databricks

```
index=main sourcetype::ProcessRollup2*
| table aid
        CommandLine
        ComputerName
        UserName
        UserSid_readable
        LocalAddressIP4
```

To get a 1:1 transformation it implies every column/field in the Search Processing Platform is in a single row in a Databricks record



How to Inject the Enrichment Logic

```python
def do_as_much_filtering_as_possible(df: DataFrame) -> DataFrame:
  return (
    df
      .where(F.expr("sourcetype LIKE 'ProcessRollup2%'"))
      .where(F.expr("term('ProcessRollup2')"))
      .where(F.expr("term('TODO_BACKSLASHES')"))
      .where(F.expr("CommandLine LIKE '%\\\\XXX\\\\XXXX\\\\XXXX\\\\XXXX\\\\XXX%'"))
      .where(F.col('_time') >= (F.expr('now()') + F.expr("INTERVAL '-1 hours'")))
      .withColumn('IOC',
        F.regexp_extract(
          F.col('_raw'),
          '(?i)(?<IOC>(YYYYYY|YYYYYY|YYYYY))', 1)
      )
  )
```

```python
def do_aggregation(df: DataFrame) -> DataFrame:
  return (
    df
    .orderBy(F.col('_time').asc())
    .groupBy('aid')
    .agg(F.count(F.lit(1)).alias('count'),
      F.first(F.col('_time'), True).alias('earliest'),
      F.last(F.col('_time'), True).alias('latest'),
      F.collect_set(F.col('ComputerName')).alias('ComputerName'))
    .withColumn('x', F.date_format(F.col('earliest'), 'askjdhlakshdjashd'))
    .select(
      F.col('aid'),
      F.col('ComputerName'),
      F.col('LocalAddressIP4'),
      F.col('FileName'),
      F.col('event_simpleName'),
      F.col('IOC'),
      F.col('count'),
      F.col('CommandLine'),
      F.col('earliest'),
      F.col('latest'),
      F.col('earliest_h'),
      F.col('latest_h'))
  )
```

```python
def uc_01():
  return (
    spark
    .read
    .table('event_processrollup2')
    .transform(do_as_much_filtering_as_possible)
    .transform(with_aidmaster)
    .transform(with_ipv4)
    .transform(do_aggregation)
    .write
    .format('delta')
    .save(somewhere)
  )
```

How to Inject the Enrichment Logic

```
def do_as_much_filtering_as_possible(df: DataFrame) → DataFrame:
  return (
    df
      .where(F.expr("sourcetype LIKE 'ProcessRollup2%'"))
      .where(F.expr("term('ProcessRollup2')"))
      .where(F.expr("term('TODO_BACKSLASHES')"))
      .where(F.expr("CommandLine LIKE '%\\\\XXX\\\\XXXX\\\\XXXX\\\\
      .where(F.col('_time') ≥ (F.expr('now()') + F.expr("INTERVAL
      .withColumn('IOC',
        F.regexp_extract(
          F.col('_raw'),
          '(?i)(?<IOC>(YYYYYY|YYYYYY|YYYYY))', 1)
      )
    )
```

```
def do_aggregation(df: DataFrame) → DataFrame:
  return (
    df
    .orderBy(F.col('_time').asc())
    .groupBy('aid')
    .agg(F.count(F.lit(1)).alias('count'),
      F.first(F.col('_time'), True).alias('earliest'),
      F.last(F.col('_time'), True).alias('latest'),
      F.collect_set(F.col('ComputerName')).alias('ComputerName'))
    .withColumn('x', F.date_format(F.col('earliest'), 'askjdhlakshdj
    .select(
      F.col('aid'),
      F.col('ComputerName'),
      F.col('LocalAddressIP4'),
      F.col('FileName'),
      F.col('event_simpleName'),
      F.col('IOC'),
      F.col('count'),
      F.col('CommandLine'),
      F.col('earliest'),
      F.col('latest'),
      F.col('earliest_h'),
      F.col('latest_h'))
  )
```


```
processrollup2')
as_much_filtering_as_possible)
n_aidmaster)
n_ipv4)
aggregation)
```

DATA+AI SUMMIT 2022

Business Outcomes

Why it matters

Abigail Shriver

Lead Cybersecurity Engineer

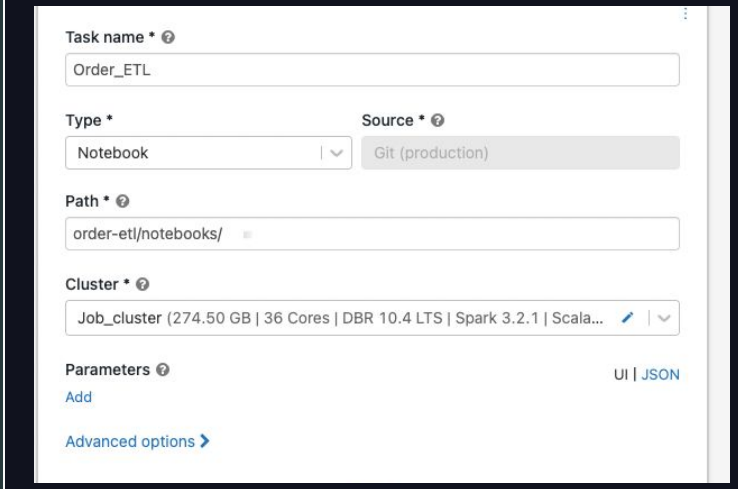
HSBC Cybersecurity Sciences & Analytics

ORGANIZED BY databricks

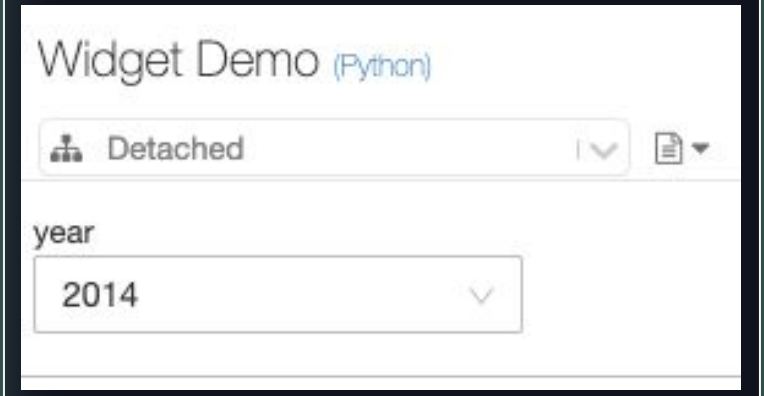
HSBC

Automation at Scale

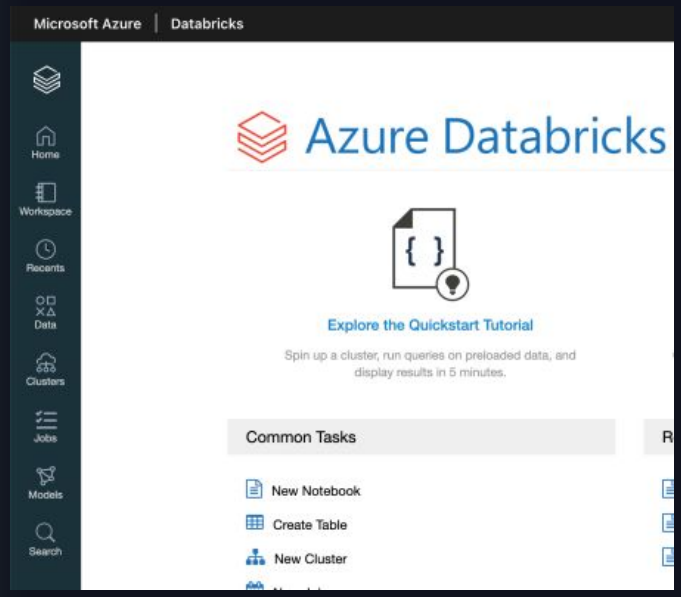
Scheduled or executed one time



Pass parameters through notebooks

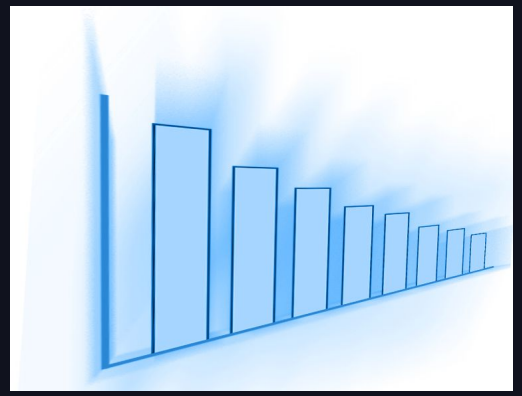


Execution



Savings

Time and Resources

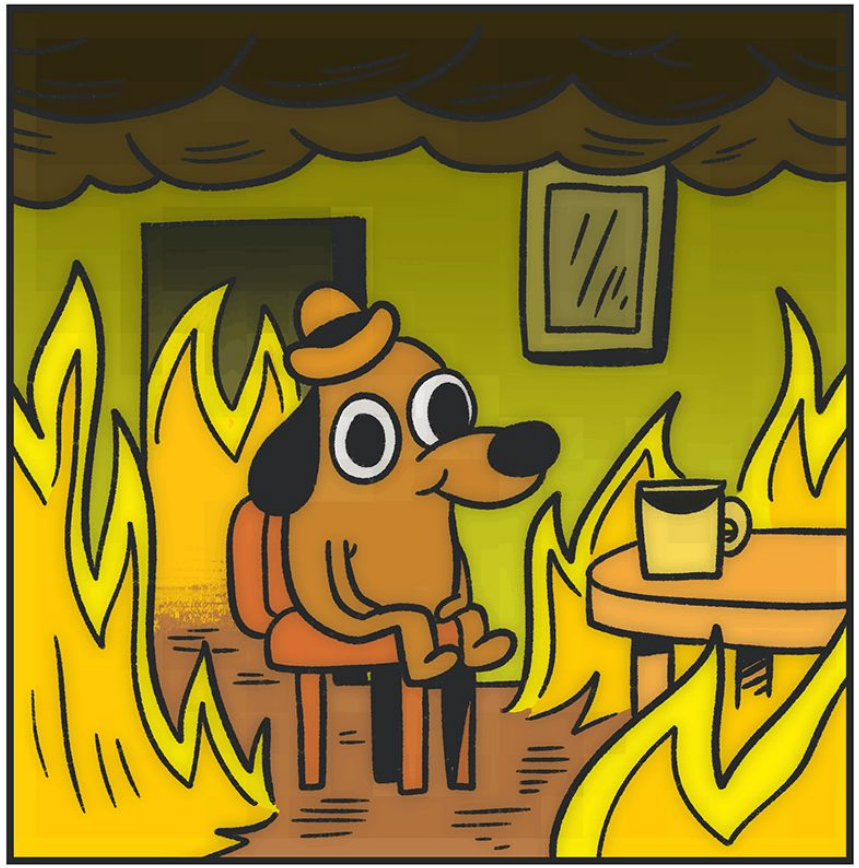


Data Storage

Compute

Historical and Scalable Metrics

Incident Response



Learning is Good

Educational

Onboarding and usability

DATA+AI
SUMMIT 2022

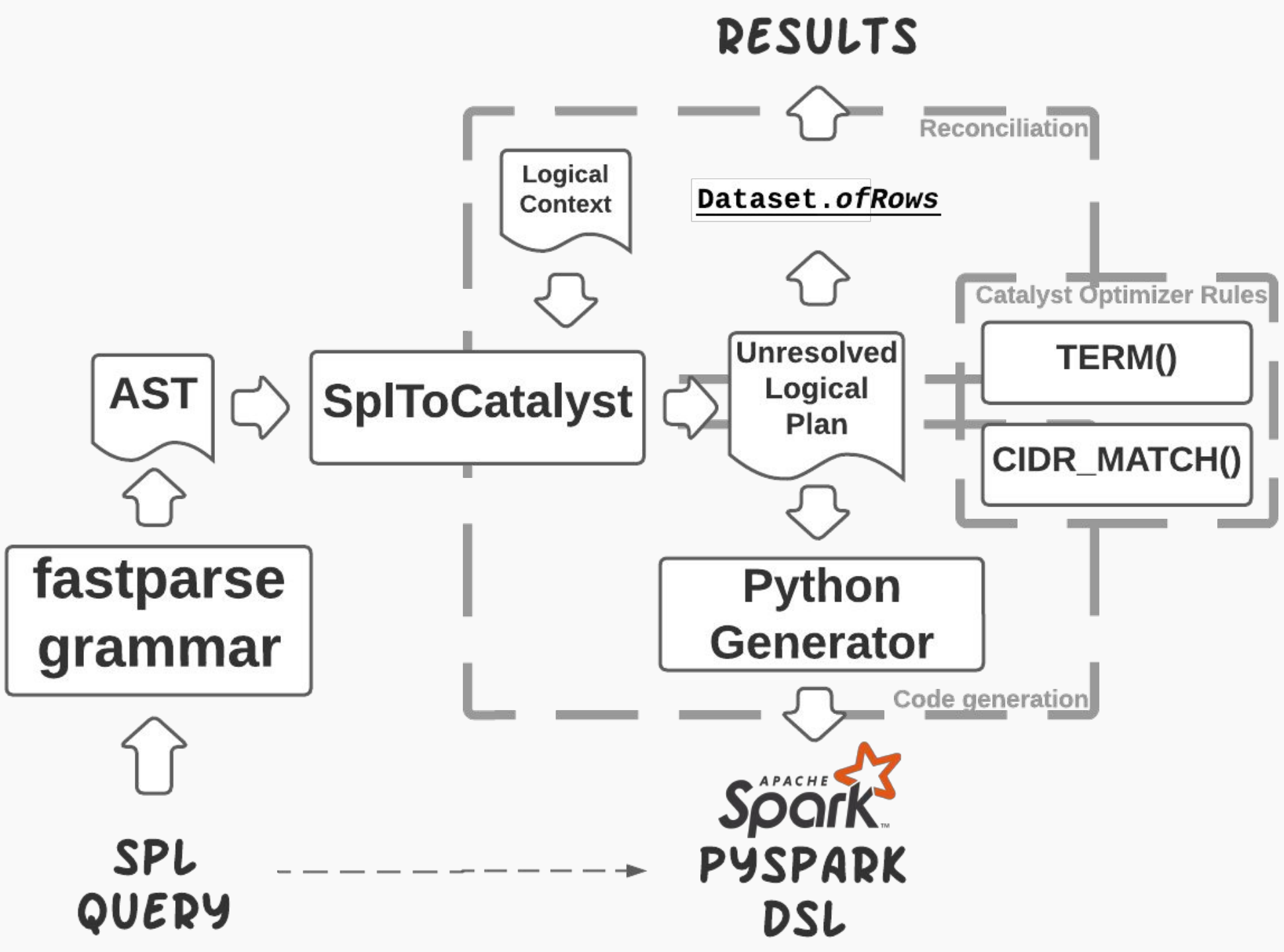
Anatomy of a translating compiler

… or how to use debugger all the time.

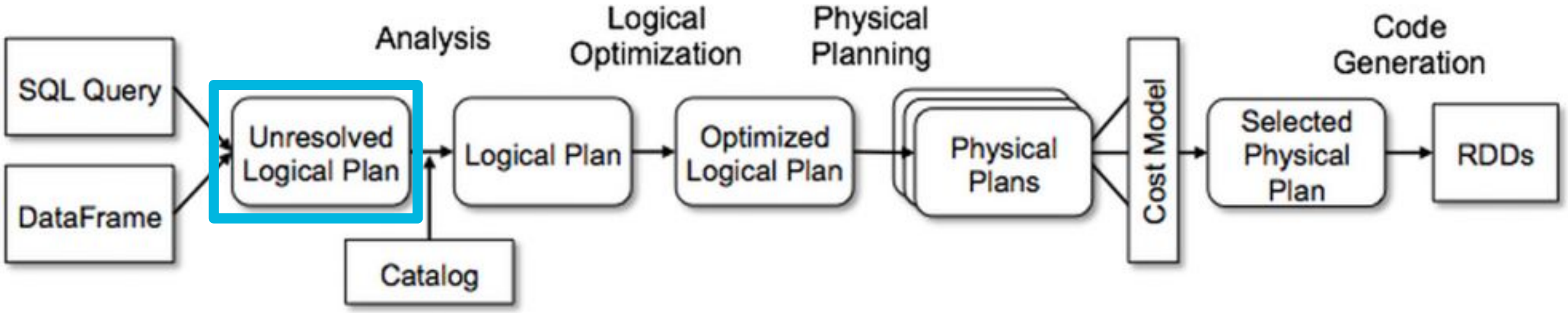
Serge Smertin
Senior Resident Solutions Architect
Databricks

ORGANIZED BY databricks

HSBC


RESULTS
Reconciliation
Logical Context
Dataset.ofRows
Catalyst Optimizer Rules
AST
SplToCatalyst
Unresolved Logical Plan
TERM()
CIDR_MATCH()
fastparse grammar
Python Generator
Code generation
APACHE Spark
PYSPARK DSL
SPL QUERY

Apache Spark query execution recap



```
code IN(4*, 5*)
```

code IN(4*, 5*)

```
def argu[_: P]: P[Expr] = termCall | call | constant
def parens[_: P]: P[Expr] = "(" ~ expr ~ ")"
def primary[_: P]: P[Expr] = unaryOf(expr) | fieldIn | parens | argu
def expr[_: P]: P[Expr] = binaryOf(primary, ALL)

def impliedSearch[_: P]: P[SearchCommand] =
  "search".? ~ expr.rep(max = 100) map(_.reduce((a, b) => Binary(a, And, b))) map SearchCommand
```

code IN(4*, 5*)

```
def argu[_: P]: P[Expr] = termCall | call | constant
def parens[_: P]: P[Expr] = "(" ~ expr ~ ")"
def primary[_: P]: P[Expr] = unaryOf(expr) | fieldIn | parens | argu
def expr[_: P]: P[Expr] = binaryOf(primary, ALL)

def impliedSearch[_: P]: P[SearchCommand] =
  "search".? ~ expr.rep(max = 100) map(_.reduce((a, b) => Binary(a, And, b))) map SearchCommand
```

```
test( testName = "code IN(4*, 5*)") {
  p(impliedSearch(_), SearchCommand(
    FieldIn("code", Seq(
      Wildcard("4*"),
      Wildcard("5*")
    ))))
}
```

```
SearchCommand(
 FieldIn("code", Seq(
   Wildcard("4*"),
   Wildcard("5*")
 )))
```

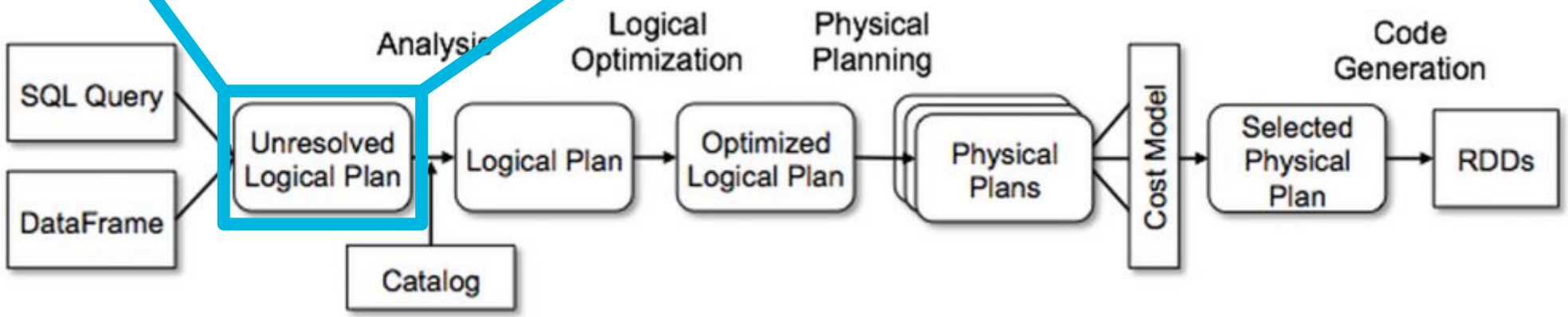
```
SearchCommand(
 FieldIn("code", Seq(
   Wildcard("4*"),
   Wildcard("5*")
 )))
```

```
Or(
  Like(UnresolvedAttribute("code"),
    Literal.create("4%"), '\\'),
  Like(UnresolvedAttribute("code"),
    Literal.create("5%"), '\\'))
```

```
SearchCommand(
  FieldIn("code", Seq(
    Wildcard("4*"),
    Wildcard("5*")
)))
```

```
private def expression(ctx: LogicalContext, expr: ast.Expr): Expression = expr match {
  case ast.FieldIn(field, exprs) if exprs.exists( .isInstanceOf[ast.Wildcard]) =>
    val expand = (expr: ast.Expr) => ast.Binary(ast.Field(field), ast.Equals, expr)
    expression(ctx, exprs.tail.foldLeft(expand(exprs.head)) {
      (left, right) => ast.Binary(left, ast.Or, expand(right))
    })
  case ast.FieldIn(field, exprs) =>
    In(UnresolvedAttribute(field), exprs.map(expression(ctx, _)))
  case ast.Binary(left, ast.Equals, ast.Wildcard(pattern)) =>
    like(ctx, left, pattern)
  case ast.Binary(left, symbol, right) => symbol match {
    case ast.Or => Or(attrOrExpr(ctx, left), attrOrExpr(ctx, right))
    // ...
  }
  // ...
}
```

```
Or(
  Like(UnresolvedAttribute("code"),
    Literal.create("4%"), '\\'),
  Like(UnresolvedAttribute("code"),
    Literal.create("5%"), '\\'))
```



```
SearchCommand(
 FieldIn("code", Seq(
   Wildcard("4*"),
   Wildcard("5*")
 )))
```

```
Or(
  Like(UnresolvedAttribute("code"),
    Literal.create("4%"), '\\'),
  Like(UnresolvedAttribute("code"),
    Literal.create("5%"), '\\'))
```

```
display(spark.table('main')
.where((F.col('code').like('4%') |
        F.col('code').like('5%'))))
```

```
SearchCommand(
 FieldIn("code", Seq(
   Wildcard("4*"),
   Wildcard("5*")
 )))
```

```
Or(
  Like(UnresolvedAttribute("code"),
    Literal.create("4%"), '\\'),
  Like(UnresolvedAttribute("code"),
    Literal.create("5%"), '\\'))
```

```
case relation: UnresolvedRelation =>
 s"spark.table(${q(relation.name)})"

private def unfoldWheres(expr: Expression): String = expr match {
 case And(left, right) => s"${unfoldWheres(left)}\n${unfoldWheres(right)}"
 case _ => s".where(${expressionCode(expr)})"
}

private def expressionCode(expr: Expression): String = expr match {
 case b: BinaryOperator =>
  val symbol = jvmToPythonOverrides.getOrElse(b.symbol, b.symbol)
  s"(${expressionCode(b.left)} $symbol ${expressionCode(b.right)})"

 case attr: UnresolvedAttribute =>
   s"F.col(${q(attr.name)})"
 case Like(col, Literal(value,   @ StringType), _) =>
   s"${expressionCode(col)}.like('$value')"
 case _ =>
   s"F.expr(${q(expr.sql)})"
}
```

```
display(spark.table('main')
.where((F.col('code').like('4%') |
        F.col('code').like('5%'))))
```

Thank you

Jude Ken-Kwofie

Principal Software Engineer,
Cybersecurity Sciences & Analytics
HSBC

jude.ken-kwofie@us.hsbc.com

Abigail Shriver

Lead Cybersecurity Engineer,
Cybersecurity Sciences & Analytics
HSBC

abigail.shriver@us.hsbc.com

Serge Smertin

Senior Resident Solutions Architect
Databricks

serge.smertin@databricks.com

cybersecurity@hsbc.com

DATA+AI SUMMIT 2022

https://databricks.com/customers/hsbc

Thank you