DATA+AI SUMMIT 2022

Automating Modernization

Configuration based automation of legacy tech to the Databricks Lake House

Jared Hillam
GTM VP, BladeBridge

Simon Eligulashvili
CoFounder, BladeBridge

ORGANIZED BY databricks

The Problem

The logic is locked in runtime systems

What makes data management solutuions sticky is that organizations have spent decades building logic on top of them. Each has bespoke metadata, code, functions, workflows, and functionalities that are locked into their world.

Failed Approaches

This problem has been hard to solve

Consulting as a Product

- No market scale
- Over the wall conversion
- Give up your metadata!
- Limited Sources/Targets

One Hit Wonders

- Singular technology
- Get stuck easily
- Not configurable enough

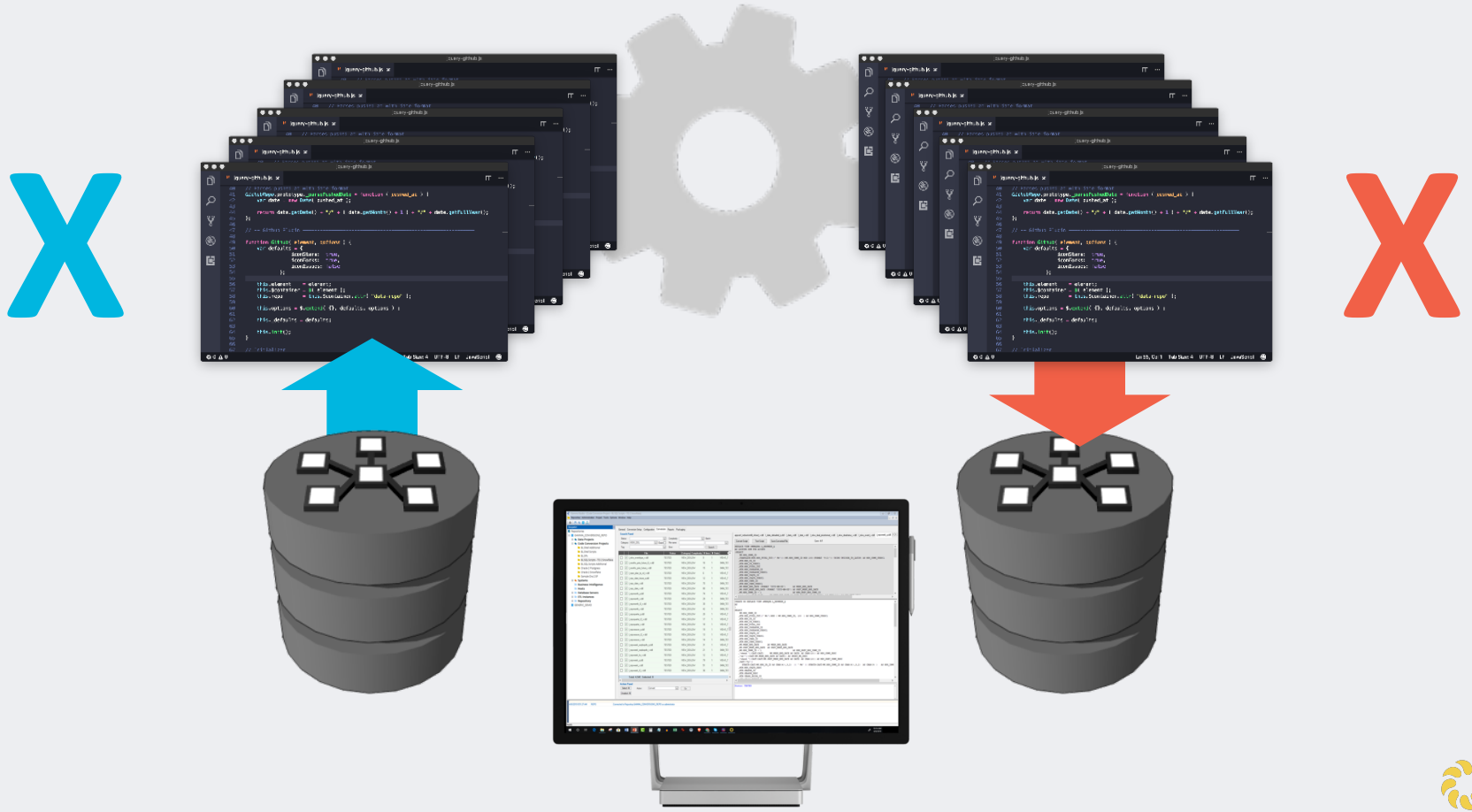
Consulting Army

- Sharded patterns
- Lost custody of logic
- People not consistent
- Lots of rework
- Not scalable

BladeBridge

What is BladeBridge Converter?

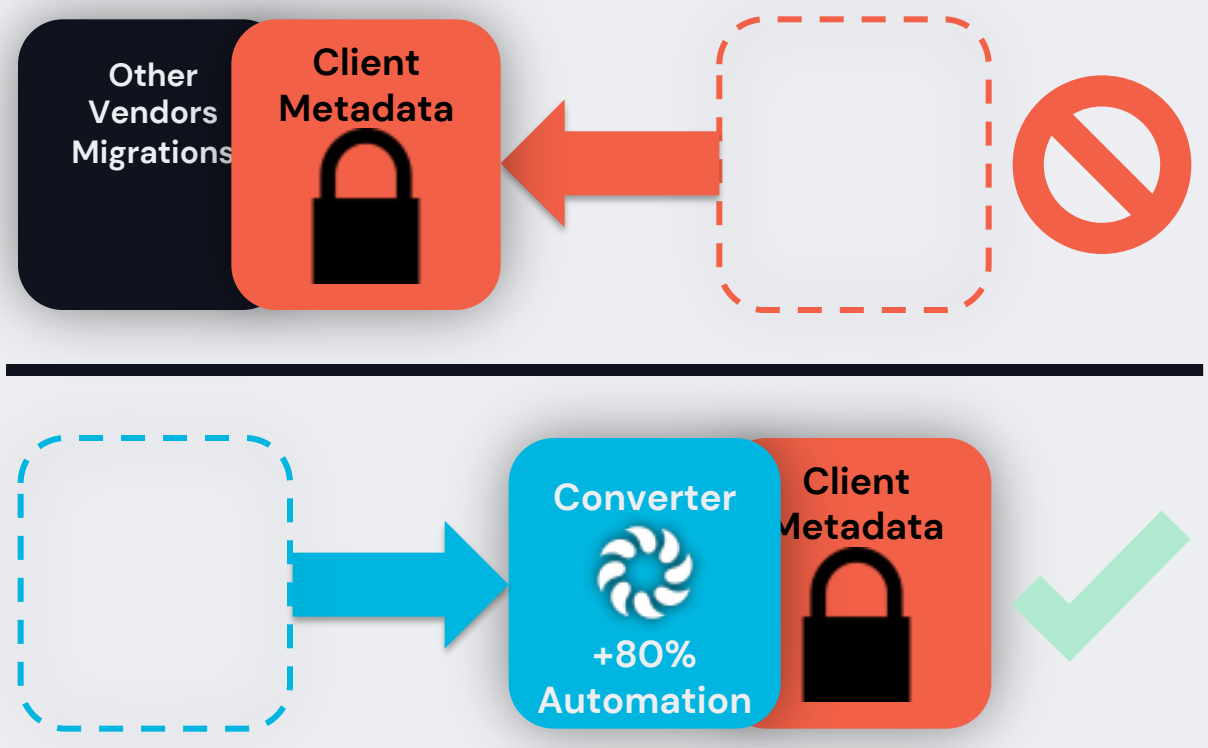
Software designed for migrating code and metadata focused on data management technologies



What makes BladeBridge different?

BladeBridge Converter is SOFTWARE

Because BladeBridge is software it can be used locally by the clients. Thus users don't have to upload their metadata to a 3rd party or service. **So, we take the tool to the code and not the code to the tool.**

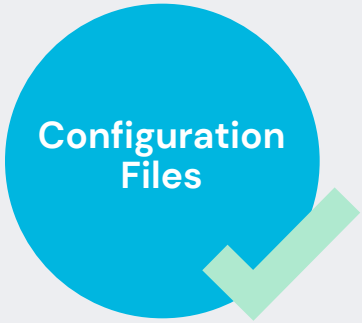


What makes BladeBridge different?

BladeBridge Converter is configuration based

Black Box

Configurable

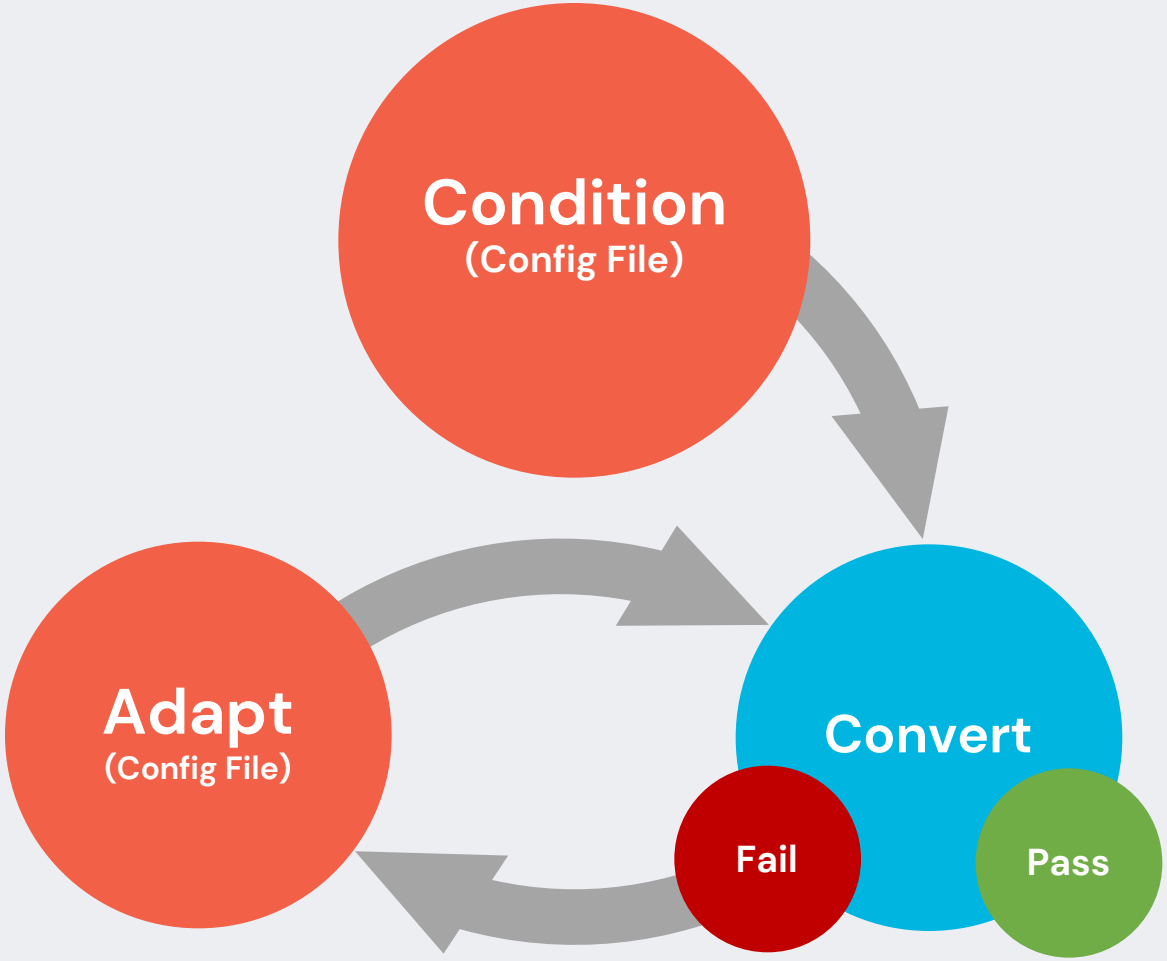


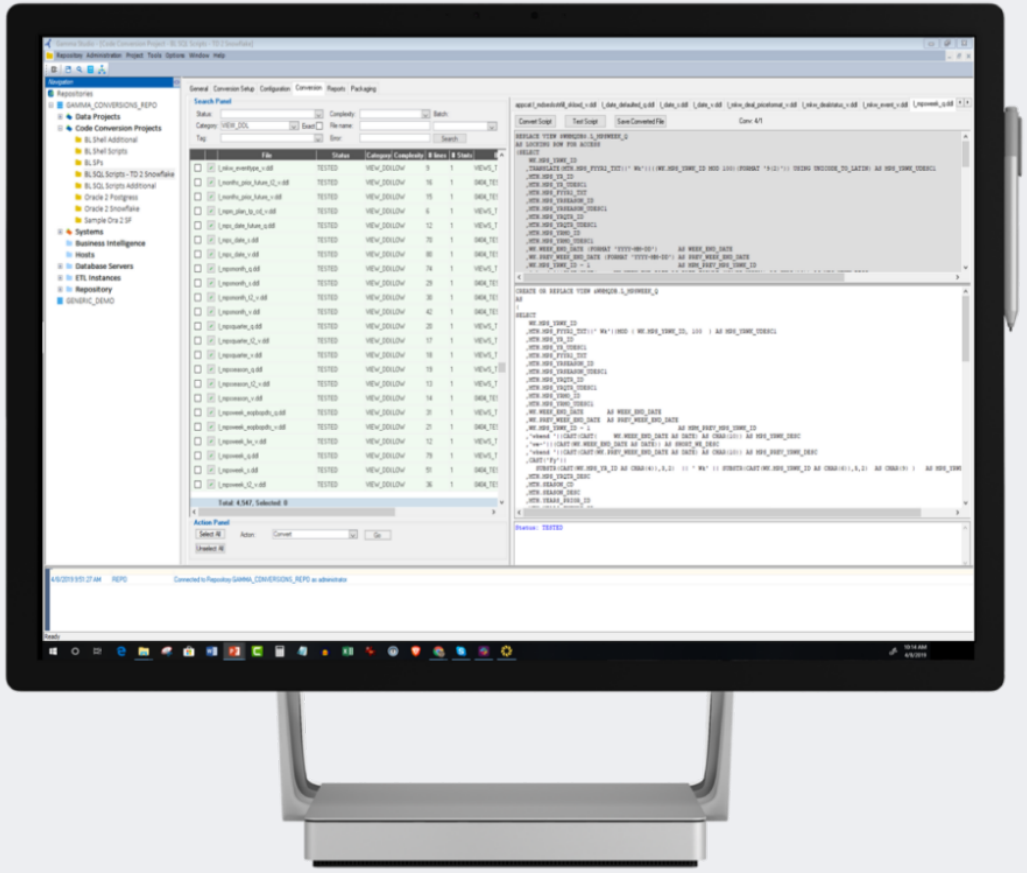
BladeBridge Converter is a code conversion engine with **externalized configuration files** which both clients and system integrators can manipulate and adapt.

Robust Community

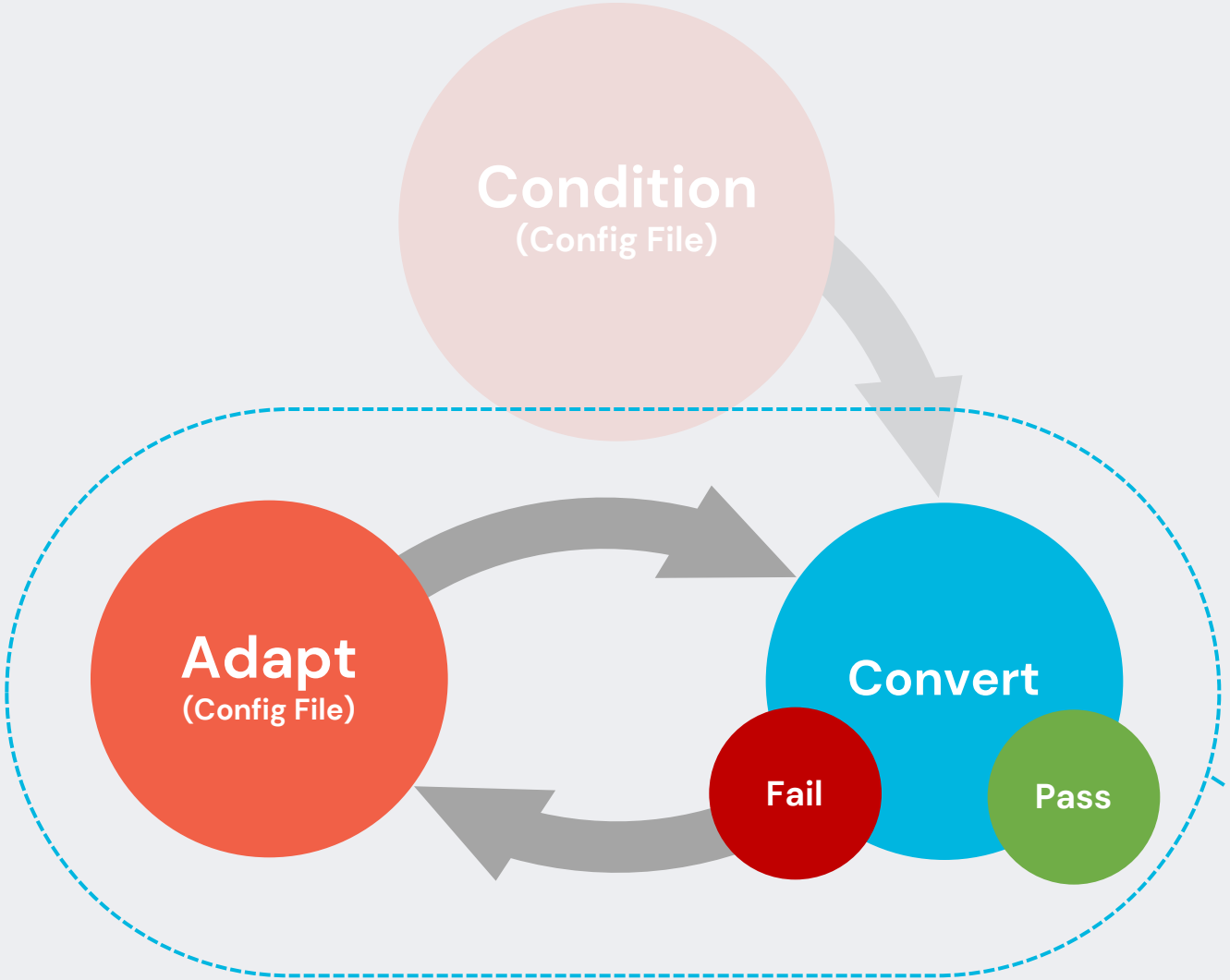
BladeBridge System Integrator Partners

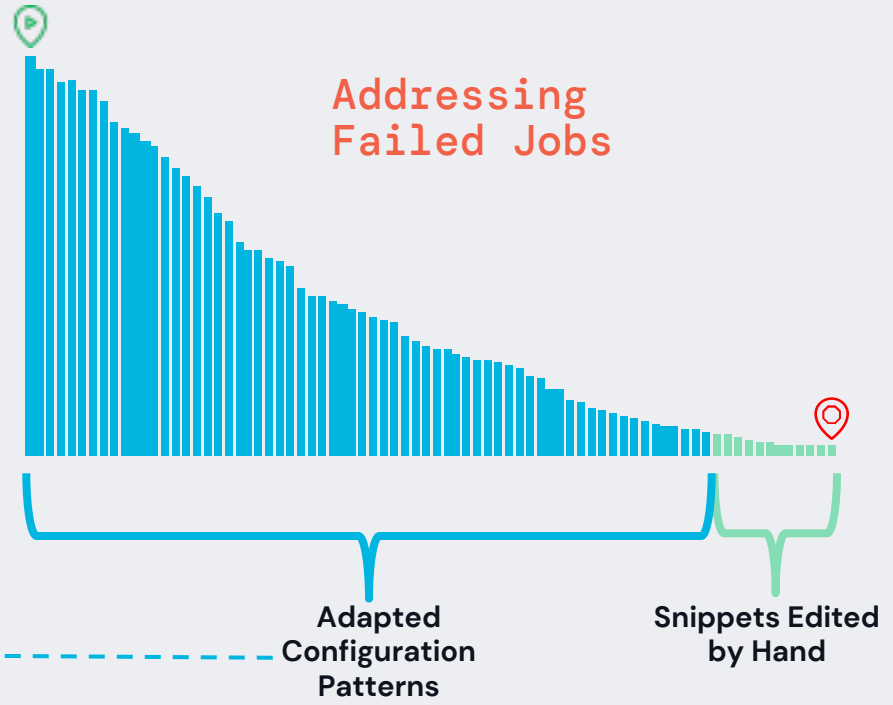
Converter Process: High Level



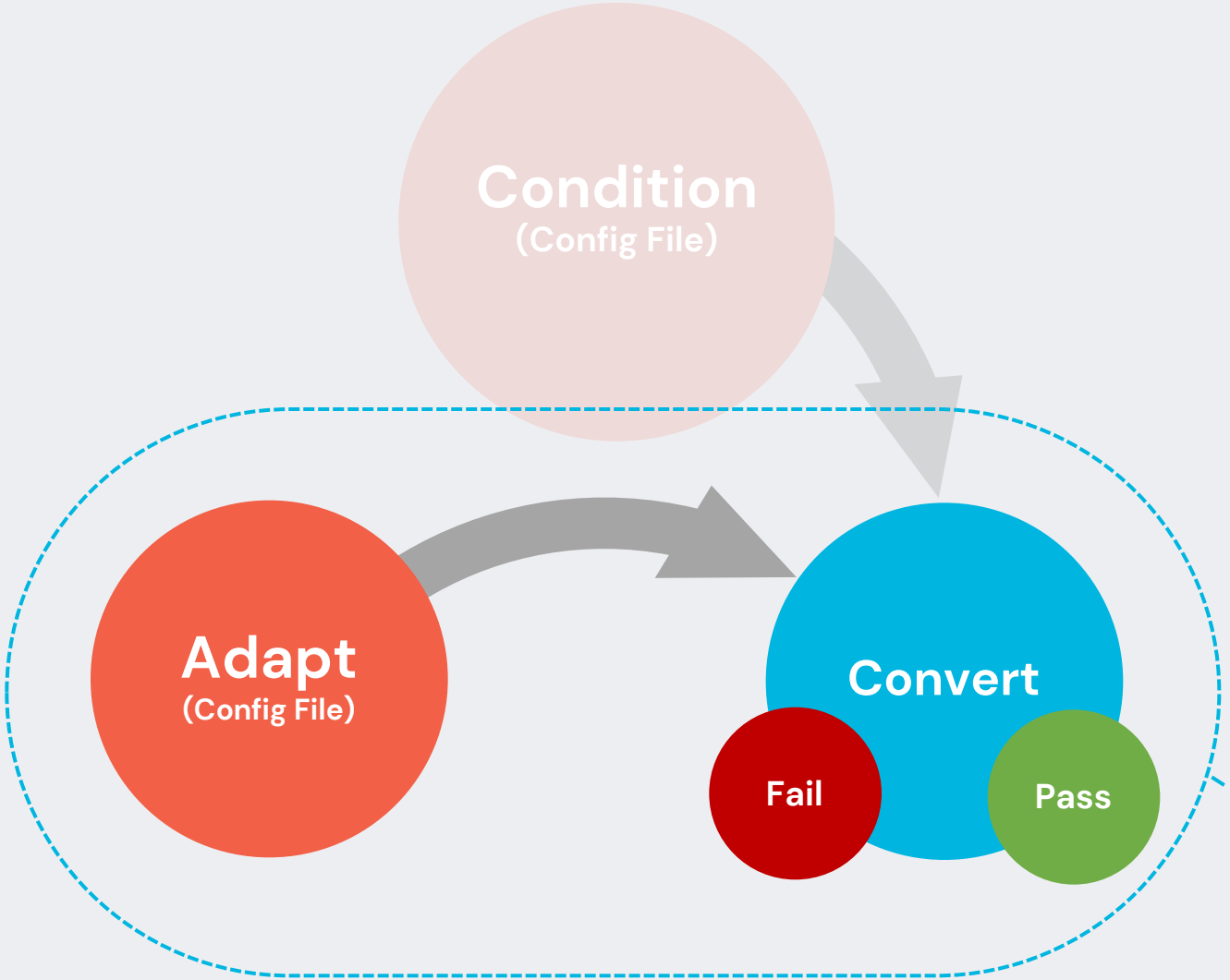


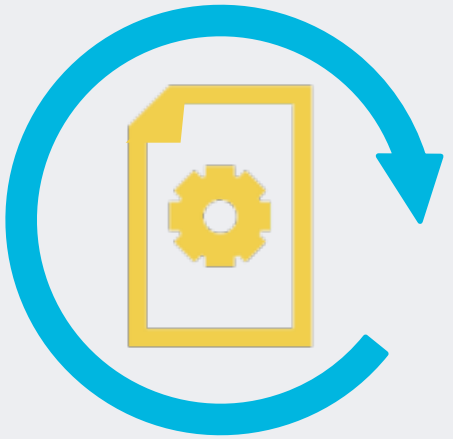
Converter Adaptation Priority



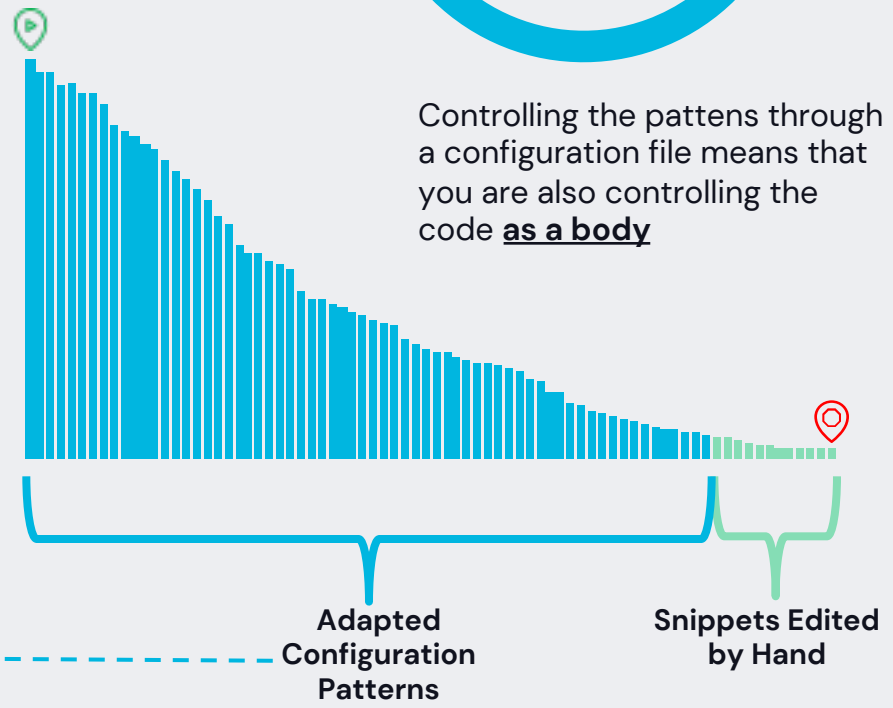


Sustaining Patterns and Custody

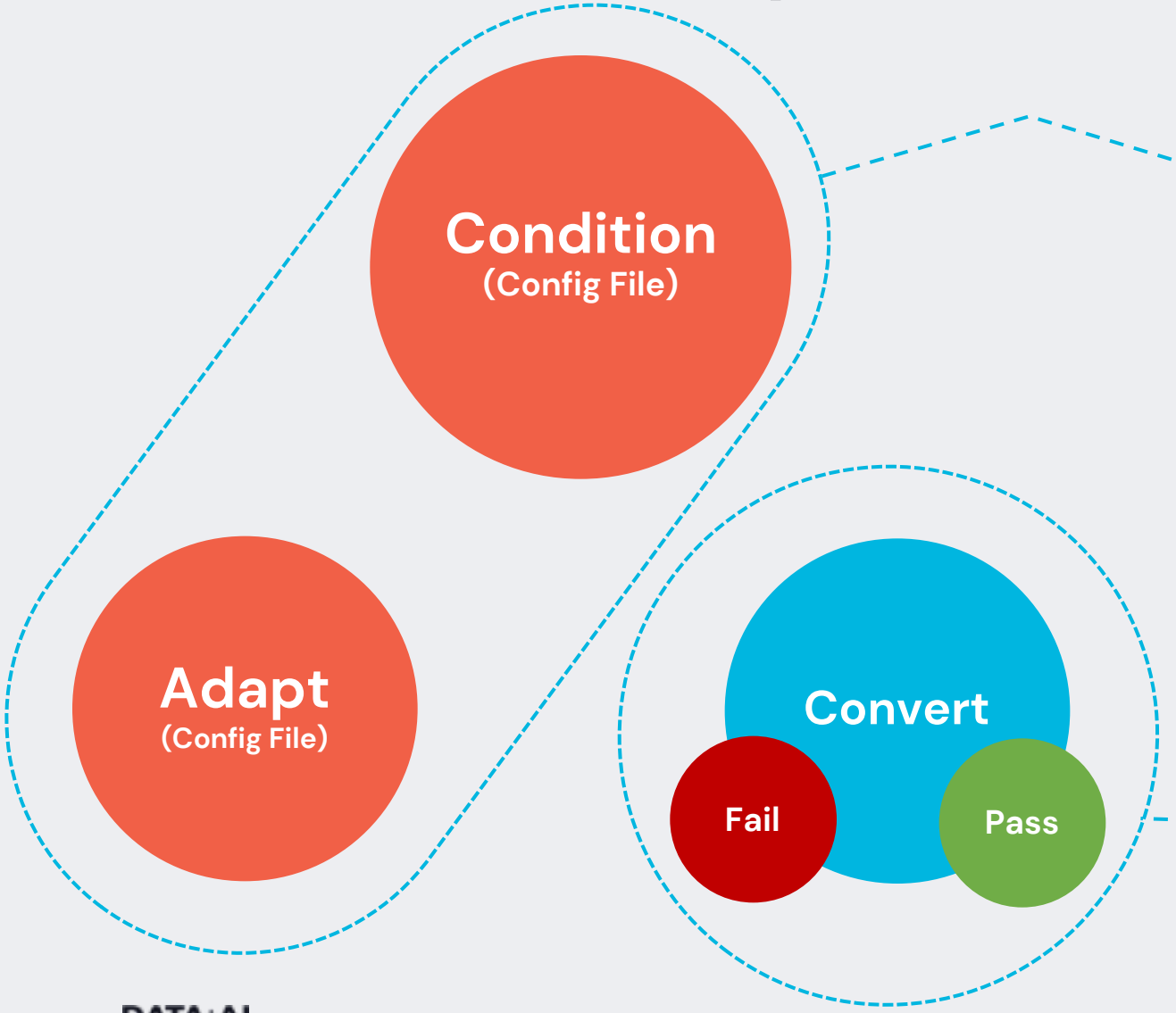




Controlling the pattens through a configuration file means that you are also controlling the code **as a body**



Core Skills Required



Conversion Architect

This resource will oversee creating and modifying patterns for the conversion project. These patterns are programed into the BladeBridge configuration files. Critical skillsets include

- Regular Expressions
- Source Metadata Knowledge
- Target Metadata Knowledge
- System Architecture Experience
- Identification and development of patterns
- Nice to have: Programming language like Python or Perl

Conversion Specialist

This resource will be executing the converter, testing the output metadata, logging the results, identifying errored patterns for the architects, conducting assessments on fixes, and suggesting patterns to Architects to configure

- Source Technology Knowledge
- Target Technology Knowledge
- Ability to debug and trace and understand error messages
- Issue resolution skills
- Identifying patterns

Configuration & Execution

Configuration Files

Loosely coupled configuration files drive BladeBridge

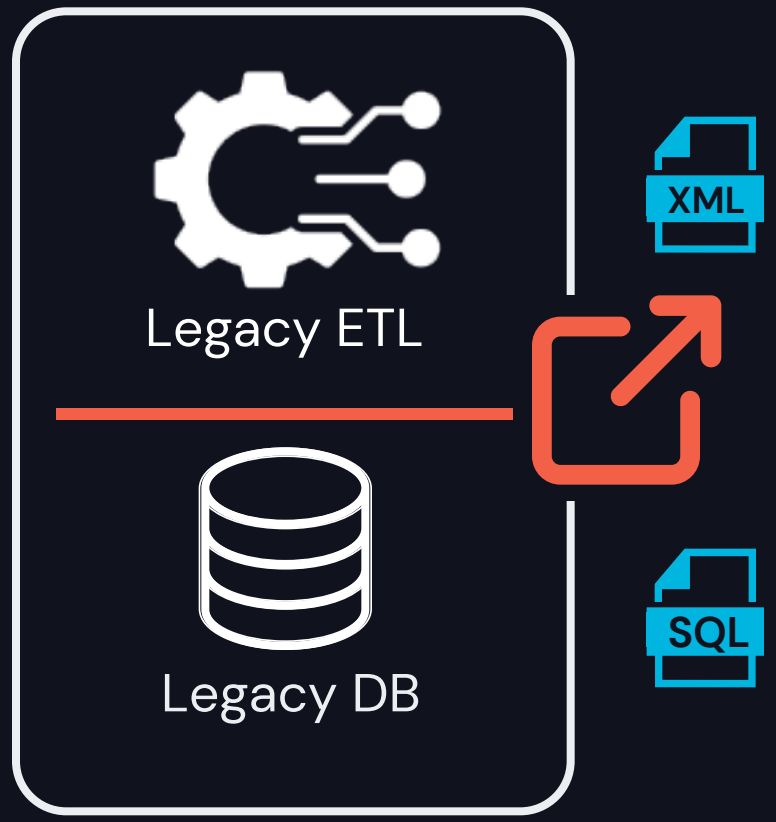
[Legacy Tech]
Reader

Bridge

[Modern Tech]
Writer

BladeBridge's Code Source

Preparing for a conversion

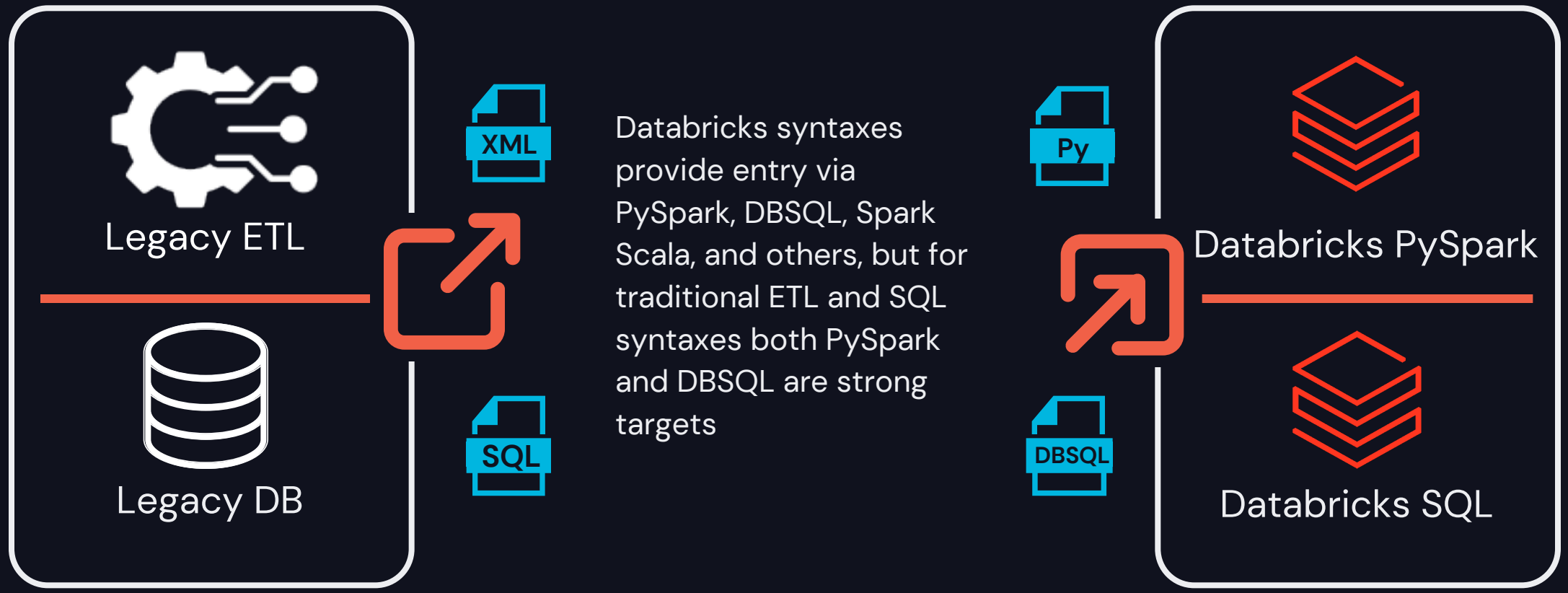


Customer or System Integrator needs to first acquire the metadata or code

- In the case of an ETL tool, it will typically be a metadata extract.
- In the case of a database, it will typically be SQL files
- In other cases, we need a representative code base of commands

Databricks target

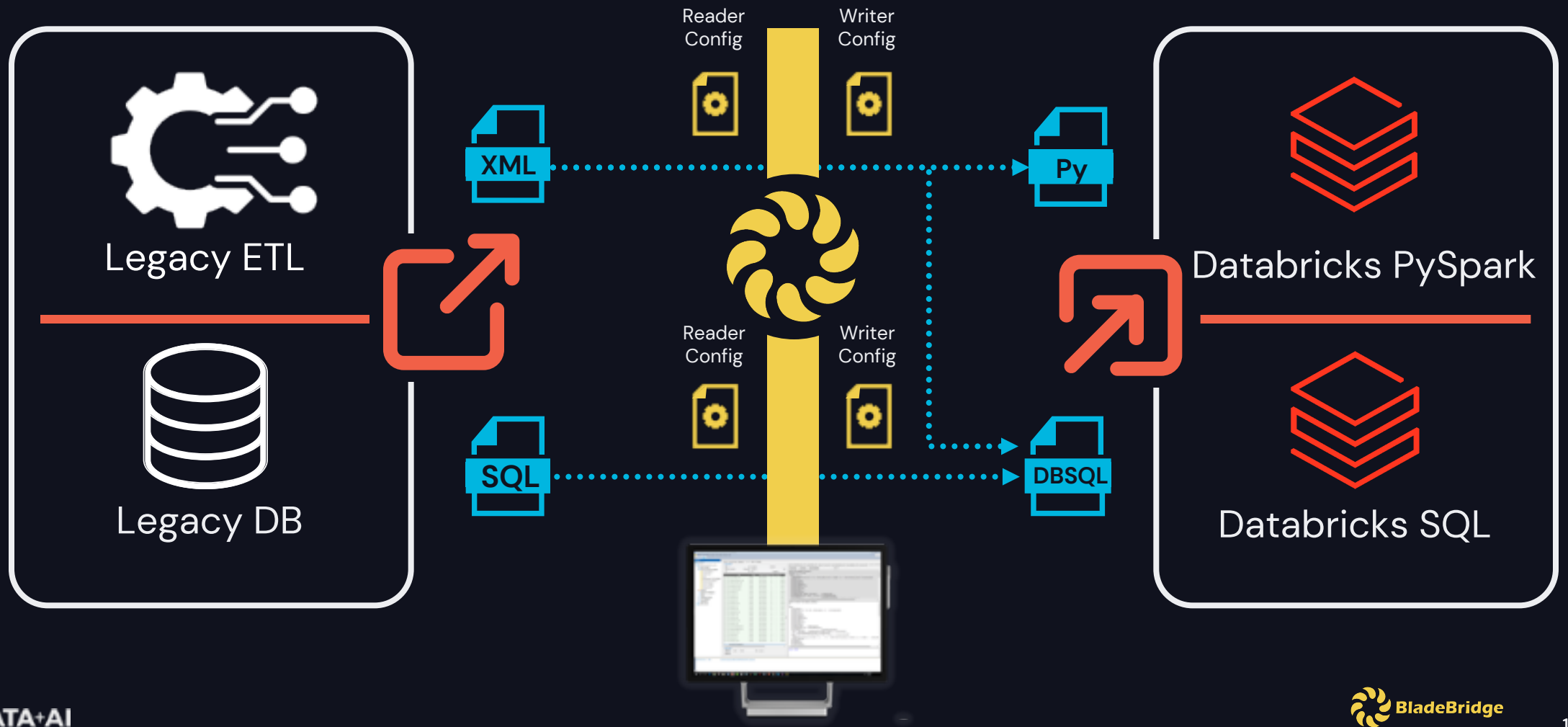
Target languages within Databricks



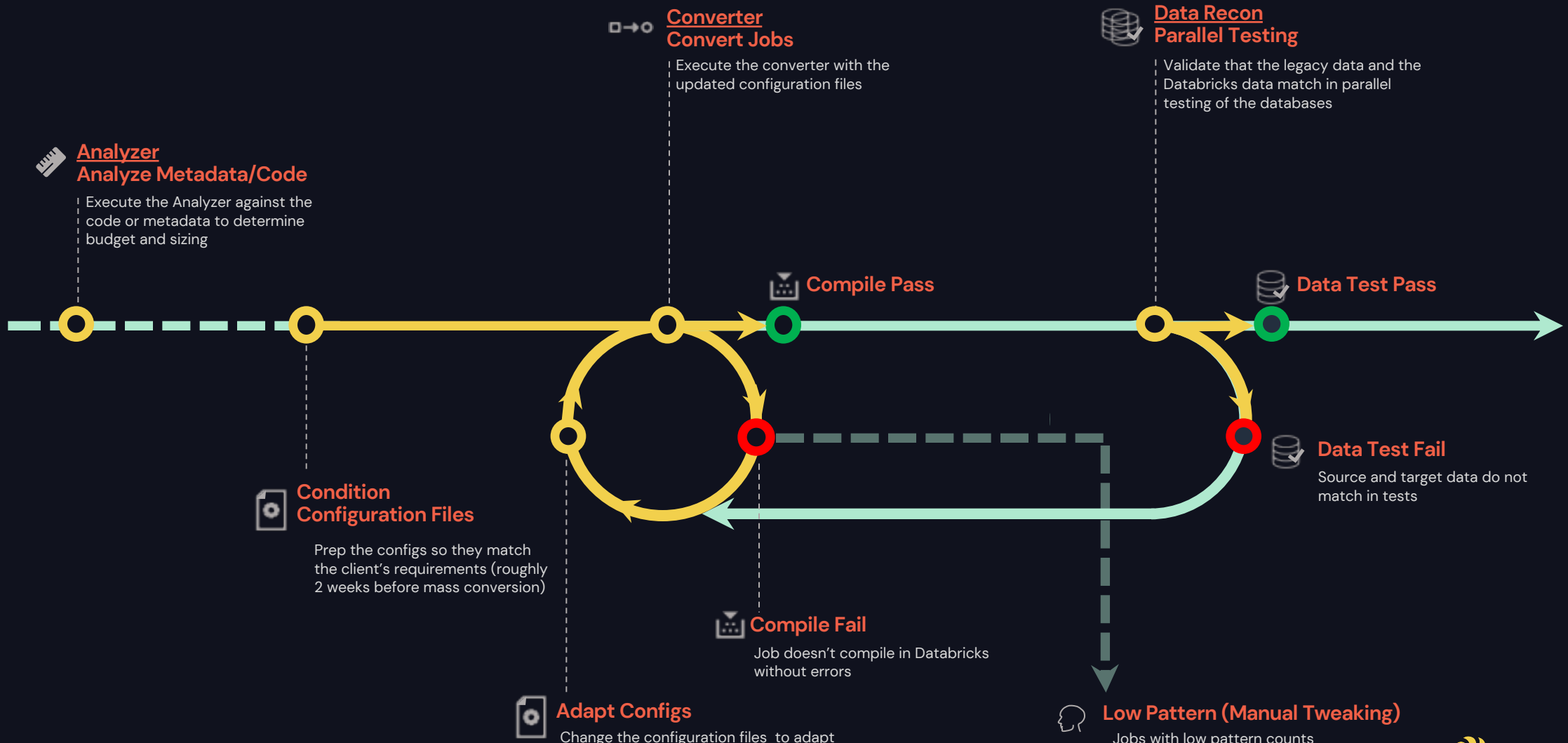
Databricks syntaxes provide entry via PySpark, DBSQL, Spark Scala, and others, but for traditional ETL and SQL syntaxes both PySpark and DBSQL are strong targets

Bridging Legacy to Databricks

Iterate with the config files



Converter Process: Detailed



Configurations

Prebuilt Configurations

Teradata

```
CREATE OR REPLACE PROCEDURE P200
(
    IN PARM1 CHAR(2) CHARACTER SET LATIN,
    IN PARM2 INTEGER
)
BEGIN

DECLARE var1 char default 'proc200';
declare var2 integer;

CREATE or replace SET TABLE TBL01 (
FNAME varchar(100),
lname varchar(100),
dob date,
start_dt date,
end_dt date,
id integer,
del_flag char(2)
)
;

insert into tbl01
    values
    ('Sally', 'Smith', '1991-01-01', '2001-05-01', '2011-09-21', 1, PARM1);

CREATE VOLATILE TABLE T_EARLY AS (
   SELECT *, NULLIFZERO(days_worked)
   FROM tbl01 t
   WHERE
      t.id < PARM2
);

select * from t_early
where months_between(start_dt, end_dt) < var2;

END;
```

Databricks DBSQL

hive_to_dbks_script1 SQL

Detached | File | Edit | View: Standard | Permissions | Run All | Clear

Cmd 1

```
-- VAR DECLARATIONS
CREATE WIDGET TEXT DBASE_TARGETTABLE DEFAULT 'RD_TBL1';
CREATE WIDGET TEXT DBASE_WORKTABLE DEFAULT 'LOAD_WORK';
CREATE WIDGET TEXT DBASE_ETTABLE DEFAULT 'LOADWORK';
CREATE WIDGET TEXT DBASE_UVTABLE DEFAULT 'LOADWORK';
CREATE WIDGET TEXT TARGETTABLE DEFAULT 'ML_TBL1';
CREATE WIDGET TEXT V_YEAR_FFSET DEFAULT '2000';

```

Cmd 2

```
--FUNCTION CALL TRANSLATIONS, VAR REFERENCES
INSERT INTO EMPLOYEE_TABLE
SELECT  STG.*,CURRENT_DATE,'OPT'
from
(
select  *,
MONTHS_BETWEEN(
    TRUNC(START_DT), END_DT - 1),
LOCATE('ADDRESS', trim(address_header))
 from   lci_load_tbls.lcixtes_eml_src_WT  where OPT_TYP_CD is not null )STG
LEFT OUTER JOIN
LCI_DW_NADR_VIEWS.LCIXTEO_EML_OPT  OPT
    ON
STG.EML_AD=OPT.EML_AD
    AND
STG.OPT_TYP_CD=OPT.OPT_TYP_CD
WHERE
STG.EML_AD IS NULL
AND EXTRACT(YEAR FROM START_DT) >= 'SV_YEAR_FFSET'
;

```

Cmd 3

```
-- TABLE DDL MANIPULATION

CREATE TABLE EMP_LIST
(
    EMP_ID INT,
    EMP_NAME STRING,
    DOB DATE
```

Prebuilt Configurations

HIVE

```
-- VAR DECLARATIONS
SET DBASE_TARGETTABLE  = 'RD_TBL1';
SET DBASE_WORKTABLE    = 'LOAD_WORK';
SET DBASE_ETTABLE      = 'LOADWORK';
SET DBASE_UVTABLE      = 'LOADWORK';
SET TARGETTABLE        = 'ML_TBL1';
SET V_YEAR_FFSET = 2000;


--FUNCTION CALL TRANSLATIONS, VAR REFERENCES
INSERT INTO EMPLOYEE_TABLE
SELECT  STG.*,CURRENT_DATE,'OPT'
from
(
select  *,
MONTHS_BETWEEN(
    TRUNC(START_DT), END_DT - 1),
LOCATE('ADDRESS', trim(address_header))
 from   lci_load_tbls.lcixtes_eml_src_WT  where OPT_TYP_CD is not null )STG
LEFT OUTER JOIN
LCI_DW_NADR_VIEWS.LCIXTEO_EML_OPT  OPT
    ON
STG.EML_AD=OPT.EML_AD
    AND
STG.OPT_TYP_CD=OPT.OPT_TYP_CD
WHERE
STG.EML_AD IS NULL
AND EXTRACT(YEAR FROM START_DT) >= ${hiveconf:V_YEAR_FFSET}
;

-- TABLE DDL MANIPULATION
CREATE TABLE EMP_LIST
(
    EMP_ID INT,
    EMP_NAME STRING,
    DOB DATE
)
    CLUSTERED BY (ID, NAME)
    SORTED BY (ID ASC)
    INTO 3 BUCKETS
    STORED AS PARQUET
;

-- FUNCTION TRANSLATIONS
INSERT INTO EMP_LIST
```

Databricks DBSQL

Prebuilt Configurations

TSQL

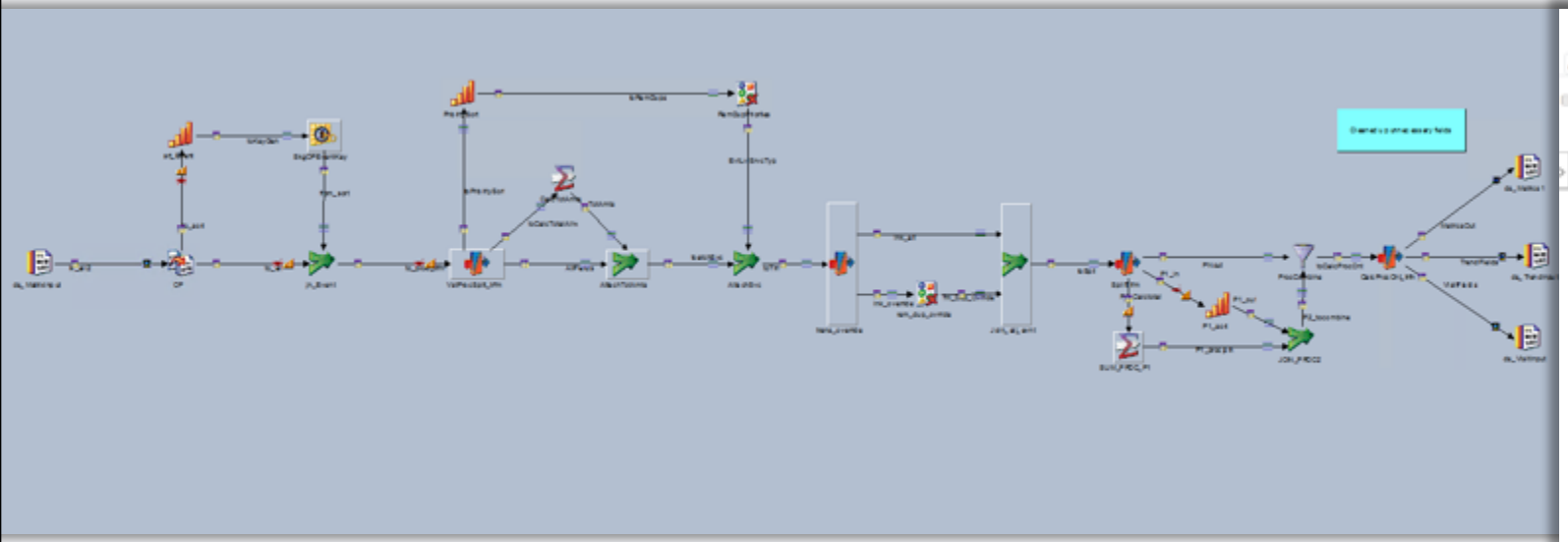
```
CREATE PROCEDURE [FACT].[SP_Populate_KPI_Values]

    @PrmStartRange DATETIME,
    @PrmEndRange DATETIME,
    @BatchLogIDOriginal INT

    WITH RECOMPILE
    AS
    BEGIN

    DECLARE @StartRange INT,
            @EndRange INT

--- Deletes last 15 days records and Reload data to get any new records which got loaded recently

    DELETE FROM [FACT].[TurbineOpHoursCounter_KPIValues_Static] WHERE [DateKey] >= CAST(REPLACE(CONVERT(CHAR(10), @PrmStartRange, 121), '-',


 --- Max Ops Counter values
    IF OBJECT_ID('FACT.TurbineOpHoursCounterKPIValues_Static_Max') IS NOT NULL
        TRUNCATE TABLE FACT.TurbineOpHoursCounterKPIValues_Static_Max

    INSERT INTO FACT.TurbineOpHoursCounterKPIValues_Static_Max
    SELECT FacilityCallSign,TurbineName
            ,MAX(TurbineOpHoursCounter) MaxTOC
            ,MAX(TurbineOpHoursCounterB) MaxTOCB
            ,MAX(TurbineOpHoursCounterR) MaxTOCR
        FROM [FACT].[TurbineOpHoursCounter_KPIValues_Static]
        GROUP BY FacilityCallSign,TurbineName ORDER BY 1


    SELECT @StartRange =  (SELECT CASE WHEN MAX(DateKey) IS NOT NULL THEN CONVERT(VARCHAR,CONVERT(DATETIME,CONVERT(CHAR(8),MAX(DateKey)))+1,
    SELECT @EndRange = CAST(REPLACE(CONVERT(CHAR(10), @PrmEndRange, 121), '-', '') AS INT)

        IF OBJECT_ID( 'Tempdb..#TurbineOpHoursCounter_KPITemp') IS NOT NULL
    DROP TABLE #TurbineOpHoursCounter_KPITemp

   ;WITH cte_MaxTurbineNames AS ( SELECT FacilityCallSign,TurbineName,MAX(MaxTOC) AS MaxTOC, MAX(MaxTOCB) AS MaxTOCB, MAX(MaxTOCB) AS MaxTOC
                    FROM FACT.TurbineOpHoursCounterKPIValues_Static_Max
                    GROUP BY TurbineName,FacilityCallSign )

 SELECT
        TurbineKey,
        EventTypeKey,
        Facilitykey ,
        XX.FacilityCallSign,
        DateKey,
        TimeKey,
        XX.TurbineName,
        [DateTime],
        EventType,
        EventCode,
        InServiceFlag,
        TurbineActivePowerkW,
        [LowPowerFilter(KW)],
        [RatedPowerFilter(KW)],
        LostProductionkWh,
        EquipmentMakeModel,
        EventFaultFlag,
  BooleanCtrALL,
    TurbineOpHoursCounter+ISNULL(MaxTOC,0) AS TurbineOpHoursCounter,
  BooleanCtrBELOW,
    TurbineOpHoursCounterB+ISNULL(MaxTOCB,0) AS TurbineOpHoursCounterB,
  BooleanCtrATRATED,
```

Databricks DBSQL

```
dbutils.widgets.text("PrmStartRange", "")
v_PrmStartRange = dbutils.widgets.get("PrmStartRange")

dbutils.widgets.text("PrmEndRange", "")
v_PrmEndRange = dbutils.widgets.get("PrmEndRange")

dbutils.widgets.text("BatchLogIDOriginal", "")
v_BatchLogIDOriginal = dbutils.widgets.get("BatchLogIDOriginal")

inputDF_lkp_Dim_eventtype = spark.read.parquet('/mnt//*')
inputDF_lkp_Dim_eventtype.createOrReplaceTempView('Dim_eventtype')
inputDF_lkp_FACT_TurbineMetric = spark.read.parquet('/mnt//*')
inputDF_lkp_FACT_TurbineMetric.createOrReplaceTempView('FACT_TurbineMetric')
inputDF_lkp_FACT_TurbineEventLog = spark.read.parquet('/mnt//*')
inputDF_lkp_DIM_Eventtype.createOrReplaceTempView('DIM_Eventtype')
inputDF_lkp_Dim_turbine = spark.read.parquet('/mnt//*')
inputDF_lkp_Dim_turbine.createOrReplaceTempView('Dim_turbine')
inputDF_lkp_Dim_facility = spark.read.parquet('/mnt//*')
inputDF_lkp_Dim_facility.createOrReplaceTempView('Dim_facility')
inputDF_lkp_DIM_Turbine = spark.read.parquet('/mnt/DIMS/Turbine/*')
inputDF_lkp_DIM_Turbine.createOrReplaceTempView('DIM_Turbine')

query_0 = spark.sql("""DELETE FROM FACT.TurbineOpHoursCounter_KPIValues_Static WHERE DateKey >= CAST(REPLACE(replace(from_unixtime(unix_timesta
--- Max Ops Counter values""".format(PrmStartRange=v_PrmStartRange))

query_1 = spark.sql("""TRUNCATE TABLE FACT.TurbineOpHoursCounterKPIValues_Static_Max""")

query_2 = spark.sql("""INSERT INTO FACT.TurbineOpHoursCounterKPIValues_Static_Max
SELECT FacilityCallSign,TurbineName
,MAX(TurbineOpHoursCounter) MaxTOC
,MAX(TurbineOpHoursCounterB) MaxTOCB
,MAX(TurbineOpHoursCounterR) MaxTOCR
FROM FACT.TurbineOpHoursCounter_KPIValues_Static
GROUP BY FacilityCallSign,TurbineName ORDER BY 1""")

StartRange_df = spark.sql("""SELECT CASE WHEN MAX(DateKey) IS NOT NULL THEN replace(from_unixtime(unix_timestamp(CAST(CAST(MAX(DateKey) AS CHAR
Fact.TurbineOpHoursCounter_KPIValues_Static WHERE MTBV_OpDays_Diff_Counter IS NULL""")
v_StartRange = StartRange_df.collect()[0][0]

EndRange_df = spark.sql("""SELECT CAST(REPLACE(replace(from_unixtime(unix_timestamp('{PrmEndRange}',"yyyy-MM-dd'T'HH:mm:ss.SSS"),"yyyy-MM-dd"),
v_EndRange = EndRange_df.collect()[0][0]

query_3 = spark.sql("""WITH cte_MaxTurbineNames AS ( SELECT FacilityCallSign,TurbineName,MAX(MaxTOC) AS MaxTOC, MAX(MaxTOCB) AS MaxTOCB, MAX(Ma
FROM FACT.TurbineOpHoursCounterKPIValues_Static_Max
GROUP BY TurbineName,FacilityCallSign )
;""")

query_4 = spark.sql("""CREATE OR REPLACE TABLE TurbineOpHoursCounter_KPITemp AS
SELECT
TurbineKey,
EventTypeKey,
TM.DateKey,
TM.TimeKey,
TM.EventTypeKey,
ET.EventType,
ET. EventCode,
TM.TurbineActivePowerkW,
RPT.LowPowerFilter(KW),
RPT. RatedPowerFilter(KW) ,
LostProductionkWh,
STUFF(EquipmentMakeModel,CHARINDEX(" ",EquipmentMakeModel),LEN(EquipmentMakeModel),'') AS EquipmentMakeModel ,
CASE
WHEN ET.EventType = 'Fault' THEN 1
ELSE 0
```

Prebuilt Configurations

DataStage



Databricks DBSQL

ProcessClaims Python

Folder \Jobs\Med, Job ProcessClaims

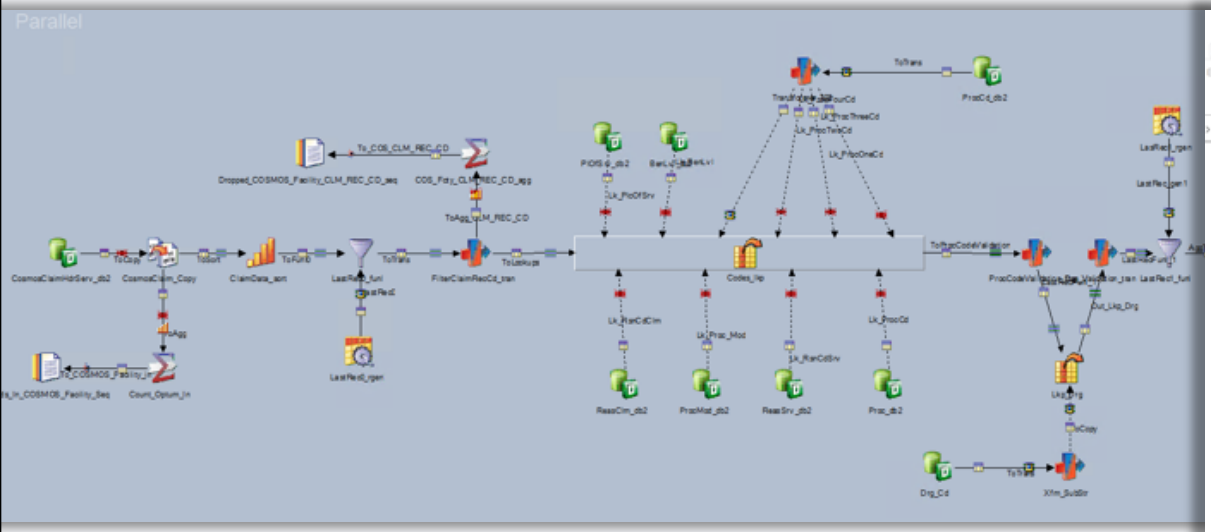
```
dbutils.widgets.text(name = "AppDir", defaultValue = "")
dbutils.widgets.text(name = "CommonCtlDir", defaultValue = "")
dbutils.widgets.text(name = "TeamDir", defaultValue = "")
dbutils.widgets.text(name = "InvID", defaultValue = "")
dbutils.widgets.text(name = "TgtDB", defaultValue = "")
dbutils.widgets.text(name = "TgtDBID", defaultValue = "")
dbutils.widgets.text(name = "TgtPswd", defaultValue = "")
dbutils.widgets.text(name = "TgtUsrID", defaultValue = "")
dbutils.widgets.text(name = "TgtSchemaOnline", defaultValue = "")
dbutils.widgets.text(name = "TgtSchemaOffline", defaultValue = "")
dbutils.widgets.text(name = "APT_CONFIG_FILE", defaultValue = "")
dbutils.widgets.text(name = "OPSSchema", defaultValue = "")
```

```
# Component lk_Demo, Type SOURCE Original node name FACT_DEMOGRAPHIC, link lk_Demo
lk_Demo = spark.sql("""
SELECT FD.MBR_SYS_ID,
EFF_DT_SYS_ID AS FACT_DOS_DT_SYS_ID,
MBR_AGE_GDR_SYS_ID,
LOB_SYS_ID,
FD.RISK_TYP_CD_SYS_ID,
ENTY_SYS_ID,
UPPER(LTRIM(RTRIM(NVL(RS.HLTH_PLN_NM, '')))) as HLTH_PLN_NM,
FD.RISK_SUPP_SYS_ID,
DM.MBR_ID as UNIQUE_MEMBER_ID,
DM.DEPN_NBR,
DM.REL_CD
,FD.PRI_PROV_SYS_ID
,FD.XCLD_FLG
,FD.MBR_SRC_SYS_ID
FROM {TgtSchemaOffline}.FACT_DEMOGRAPHICS FD
INNER JOIN {TgtSchemaOffline}.DIM_MEMBER DM
ON FD.MBR_SYS_ID=DM.MBR_SYS_ID
AND FD.ENTY_ID=DM.ENTY_ID
INNER JOIN {TgtSchemaOffline}.DIM_RISK_TYPE DRT
ON FD.RISK_TYP_CD_SYS_ID = DRT.RISK_TYP_CD_SYS_ID
where FD.ENTY_ID='{AppDir}'
""".format(AppDir=dbutils.widgets.get('AppDir'), TgtSchemaOffline=dbutils.widgets.get('TgtSchemaOffline')))
lk_Demo.createOrReplaceTempView("lk_Demo")
```

```
# Component fm_SvcTyp, Type SOURCE Original node name odbcConn_SvcTyp, link fm_SvcTyp
fm_SvcTyp = spark.sql("""
SELECT HLTH_PLN_SRVC_TYP_CD_SYS_ID, HLTH_PLN_SRVC_TYP_CD, HCE_SRVC_TYP_DESC,  HLTH_PLN_SRVC_TYP_LVL_4_NM, HLTH_PLN_SRVC_TYP_LVL_
```

Prebuilt Configurations

DataStage



Databricks PySpark

Policies Python

Folder \Jobs\Pol

Job PoliciesProcess

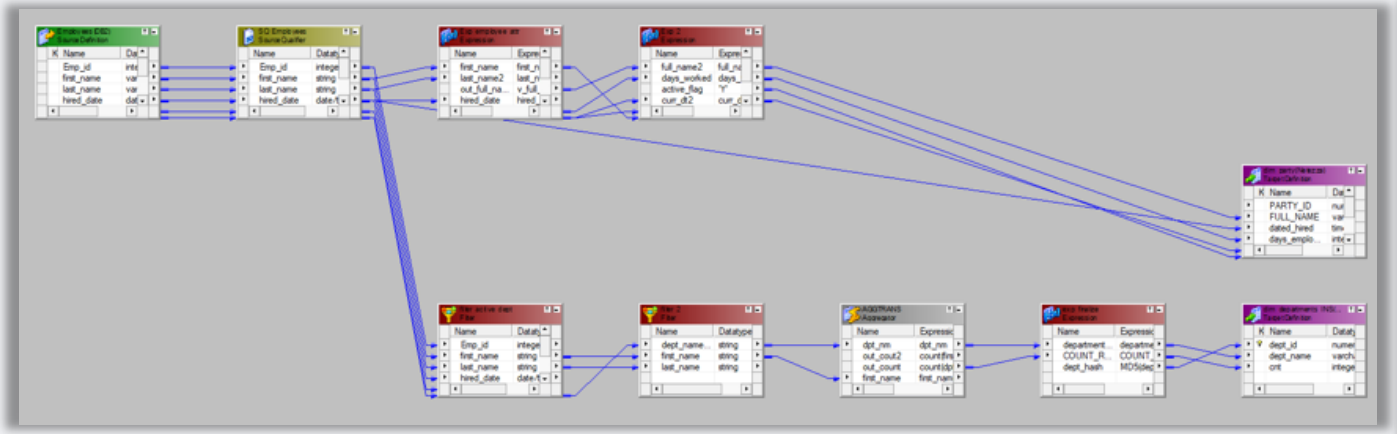
```
# Processing node Lk_BenLvl, type SOURCE
# COLUMN COUNT: 2
# Original node name BenLvl_db2, link Lk_BenLvl

Lk_BenLvl = spark.read.jdbc('DATA_SOURCE', f"""SELECT BEN_LVL_SYS_ID, BEN_PAY_CD FROM {os.environ['dbschema2']}.BENEFIT_LEVEL""")
```

```
# Processing node ToAgg_CLM_REC_CD, type TRANSFORMATION
# COLUMN COUNT: 2
# Original node name FilterClaimRecCd_tran, link ToAgg_CLM_REC_CD

ToAgg_CLM_REC_CD = ToTrans.withColumn("DF_ROW_ID", monotonically_increasing_id()) \
    .withColumn("mbrage", when(ToTrans.MBR_MM > lit(0) , ToTrans.MBR_YY).otherwise(when(ToTrans.MBR_MM == lit(0) & ToTrans.MBR_DD >= lit(0) , ToTrans.MBR_YY , ToTrans.MBR_YY - lit(1)))) \
    .withColumn("CurrSiteCd", ToTrans.SITE_CD) \
    .withColumn("CurrServDt", ToTrans.ERLY_SRVC_DT) \
    .withColumn("CurrClmAudNbr", ToTrans.CLM_AUD_NBR) \
    .withColumn("InRow", monotonically_increasing_id()) \
    .withColumn("WriteFlag", when(InRow > lit(1) &((CurrSiteCd != lag(CurrSiteCd).over(Window.orderBy('DF_ROW_ID'))) |(CurrClmAudNbr != lag(CurrClmAudNbr).over(Window.orderBy('DF_ROW_ID'))) |(CurrServDt != lag(CurrServDt).over(Window.orderBy('DF_ROW_ID')))) , lit(1)).otherwise(lit(0))) \
    .withColumn("RecFlag", when(InRow == lit(1) , ToTrans.CLM_REC_CD).otherwise(when(WriteFlag == lit(1) , ToTrans.CLM_REC_CD , RecFlag))) \
    .withColumn("PrevSiteCd", CurrSiteCd) \
    .withColumn("PrevServDt", CurrServDt) \
    .withColumn("PrevClmAudNbr", CurrClmAudNbr).select( \
    ToTrans.OPTUM_SEG_ID.alias("OPTUM_SEG_ID"), \
    ToTrans.MBR_SYS_ID.alias("MBR_SYS_ID") \
) \
.filter("(RecFlag <> \"01\" and COS_CLM_HEAD_SYS_ID <> -99) and (OPTUM_SEG_ID <> \"XXXXXXXX\")")
```

```
# Processing node ToLookups, type TRANSFORMATION
# COLUMN COUNT: 65
# Original node name FilterClaimRecCd_tran, link ToLookups

ToLookups = ToTrans.withColumn("DF_ROW_ID", monotonically_increasing_id()) \
    .withColumn("mbrage", when(ToTrans.MBR_MM > lit(0) , ToTrans.MBR_YY).otherwise(when(ToTrans.MBR_MM == lit(0) & ToTrans.MBR_DD >= lit(0) , ToTrans.MBR_YY , ToTrans.MBR_YY - lit(1)))) \
    .withColumn("CurrSiteCd", ToTrans.SITE_CD) \
    .withColumn("CurrServDt", ToTrans.ERLY_SRVC_DT) \
    .withColumn("CurrClmAudNbr", ToTrans.CLM_AUD_NBR) \
    .withColumn("InRow", monotonically_increasing_id()) \
    .withColumn("WriteFlag", when(InRow > lit(1) &((CurrSiteCd != lag(CurrSiteCd).over(Window.orderBy('DF_ROW_ID'))) |(CurrClmAudNbr != lag(CurrClmAudNbr).over(Window.orderBy('DF_ROW_ID'))) |(CurrServDt != lag(CurrServDt).over(Window.orderBy('DF_ROW_ID')))) , lit(1)).otherwise(lit(0))) \
    .withColumn("RecFlag", when(InRow == lit(1) , ToTrans.CLM_REC_CD).otherwise(when(WriteFlag == lit(1) , ToTrans.CLM_REC_CD , RecFlag))) \
    .withColumn("PrevSiteCd", CurrSiteCd) \
```

Prebuilt Configurations

Informatica



Databricks PySpark

m_employees_load Python

Folder Conversion_From_Infa, Job m_employees_load

This is a demo mapping

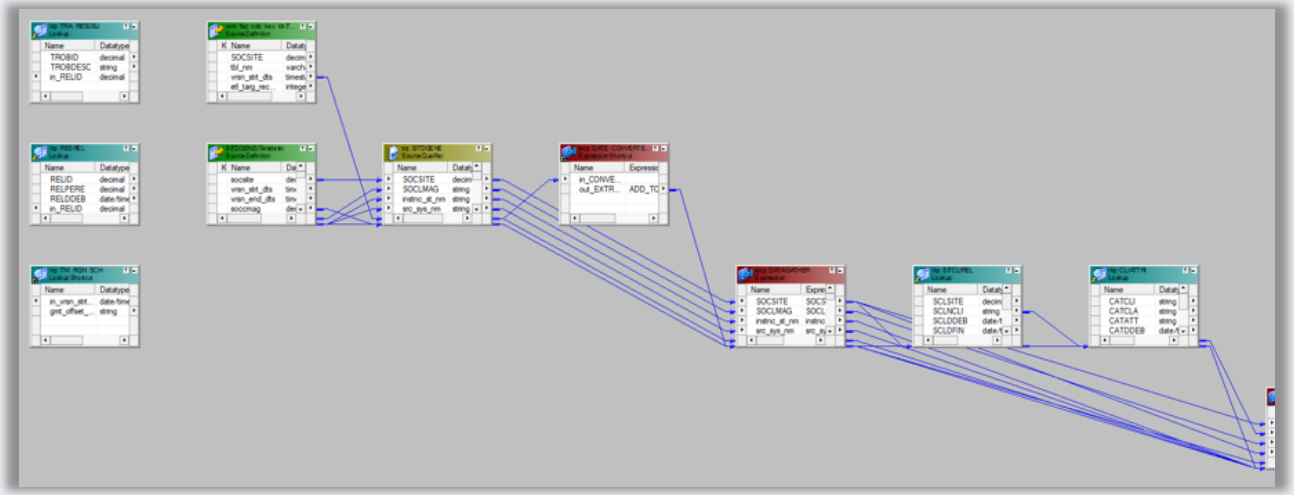
```
# Processing node SQ_Employees, type SOURCE
# COLUMN COUNT: 7
SQ_Employees = spark.read.jdbc(os.environ.get("HR_Database_CONNECT_STRING"), """SELECT
Employees.Emp_id,
Employees.first_name,
Employees.last_name,
Employees.hired_date,
Employees.last_upd_date,
Employees.salary,
Employees.dept_name
FROM Employees
WHERE Employees.hired_date > '2020-01-01'""", properties={'user': os.environ.get("HR_Database_LOGIN"), 'password': os.environ.get("HR_Database_PASSWORD"),
```

```
# Processing node filter_active_dept, type FILTER
# COLUMN COUNT: 7
filter_active_dept = SQ_Employees.select( \
    SQ_Employees.Emp_id.alias('Emp_id'), \
    SQ_Employees.first_name.alias('first_name'), \
    SQ_Employees.last_name.alias('last_name'), \
    SQ_Employees.hired_date.alias('hired_date'), \
    SQ_Employees.last_upd_date.alias('last_upd_date'), \
    SQ_Employees.salary.alias('salary'), \
    SQ_Employees.dept_name.alias('dept_name')) \
    .filter("substring(dept_name, 1, 3) != '000' and first_name != os.environ.get('VAR100')").withColumn("sys_row_id", monotonically_increasing_id())
```

```
# Processing node Exp_employee_attr, type EXPRESSION
# COLUMN COUNT: 7
Exp_employee_attr = SQ_Employees.select( \
    SQ_Employees.sys_row_id.alias('sys_row_id'), \
    SQ_Employees.first_name.alias('first_name'), \
    SQ_Employees.last_name.alias('last_name2'), \
    SQ_Employees.hired_date.alias('hired_date')).withColumn("v_full_name", concat(col('first_name') , lit(' ') , col('last_name2'))).select( \
    (col('sys_row_id')).alias('sys_row_id'), \
    col('first_name'), \
    col('last_name2'), \
    (col('v_full_name')).alias('out_full_name'), \
    col('hired_date'), \
    (datediff(current_date() , col('hired_date'))).alias('out_days_worked'), \
    (lit('Y')).alias('active'), \
    (current_date()).alias('curr_dt') \
)
```

Prebuilt Configurations

Informatica



Databricks DBSQL

m_DM_DEMO_LOAD1 Python

Folder Conversion_From_Infa, Job m_DM_DEMO_LOAD1

This is a demo mapping

Variable declaration section

```
dbutils.widgets.text("SRC_ROW_CN","""")
dbutils.widgets.text("SRC_SUM_AM","""")
dbutils.widgets.text("SESSION_RUN_ID","""")
dbutils.widgets.text("WORKFLOW_RUN_ID","""")
dbutils.widgets.text("SRC_INTF_VRSN","""")
dbutils.widgets.text("TARG_ROW_CN","""")
dbutils.widgets.text("TARG_SUM_AM","""")
dbutils.widgets.text("SRC_SUM_FIELDS","""")
dbutils.widgets.text("NALA_DM_VMOD","""")
dbutils.widgets.text("INPUT_FILE_NAME","""")
dbutils.widgets.text("SRC_SYS_NM","""")
dbutils.widgets.text("NALA_SHA_DD","""")
```

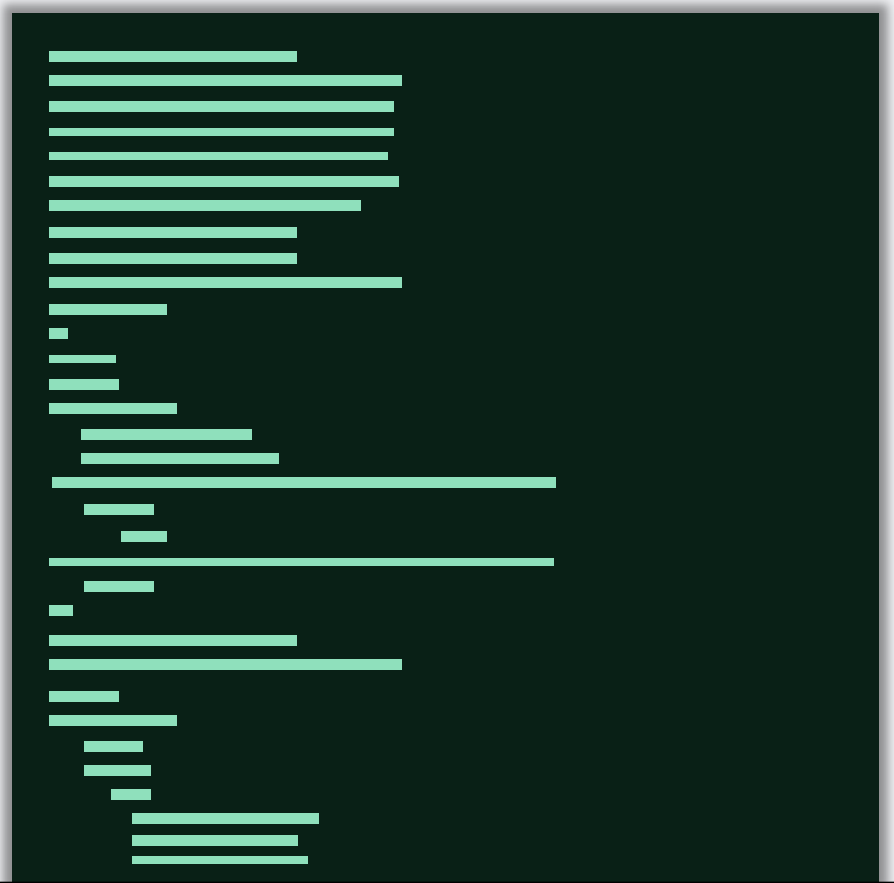
```
#Component exp_DATE_CONVERTER_GMT, Type EXPRESSION
exp_DATE_CONVERTER_GMT = spark.sql("""
SELECT
DATE_ADD(cast(SUBSTR ( LTRIM ( v_lkp_GMT ) , 2 , 1 ) as int), vrsn_strt_dts) as out_EXTRACT_DT,
query_1.source_record_id
FROM (
SELECT
LKP_1.gmt_offset_ta_intrvl /* replaced lookup LKP_TM_RGN_SCH */ as v_lkp_GMT,
* FROM sq_SITDGENE
) query_1
""")
exp_DATE_CONVERTER_GMT.createOrReplaceTempView("exp_DATE_CONVERTER_GMT")
```

```
#Component exp_DATAGATHER, Type EXPRESSION
exp_DATAGATHER = spark.sql("""
SELECT
SOCSITE as SOCSITE,
SOCLMAG as SOCLMAG,
instnc_st_nm as instnc_st_nm,
src_sys_nm as src_sys_nm,
vrsn_strt_dts as vrsn_strt_dts,
etl_targ_rec_set_id as etl_targ_rec_set_id,
out_EXTRACT_DT as in_vrsn_strt_dts_EST,
to_date ( CHAR(v_vrsn_strt_dts_EST, 'YYYY-MM-DD HH24:MI:SS') , 'YYYY-MM-DD HH24:MI:SS' ) as out_vrsn_strt_dts_EST,
SOCCMAG as SOCCMAG,
SOCZGEO as SOCZGEO,
```

New Configurations

Usually 5-20 Days

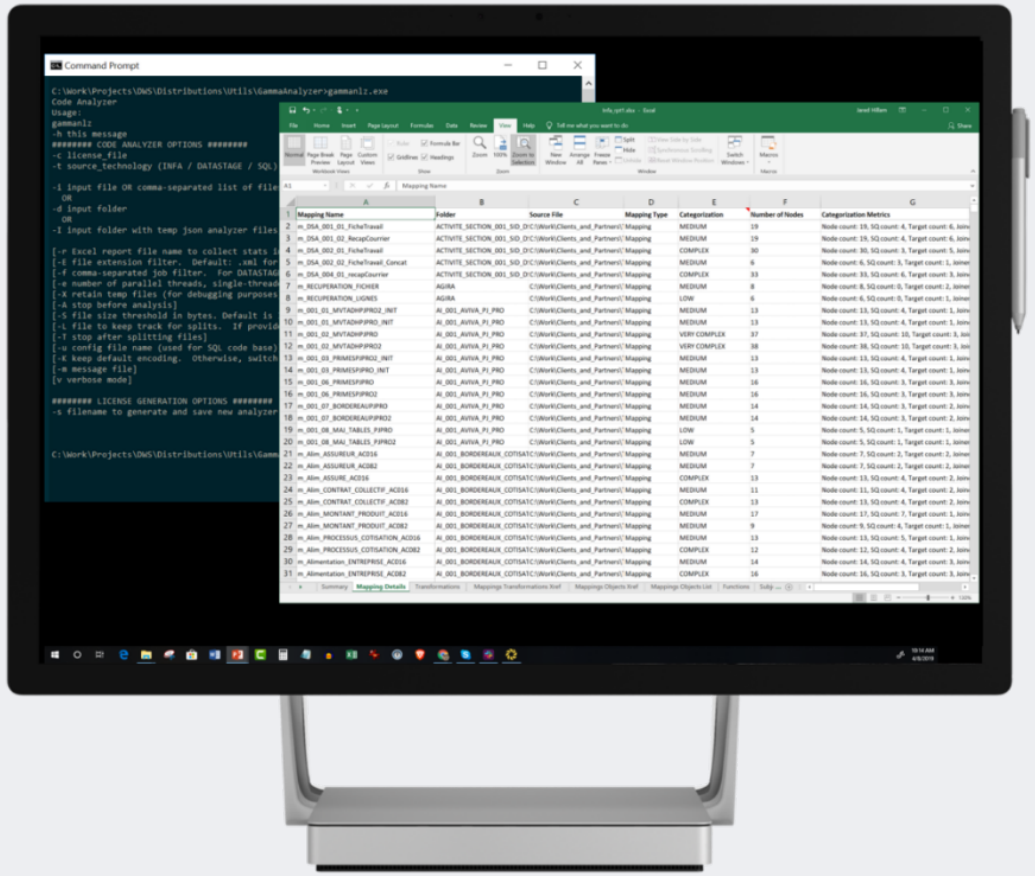
[Legacy Tech]



Databricks

How much does it Cost?

BladeBridge Analyzer



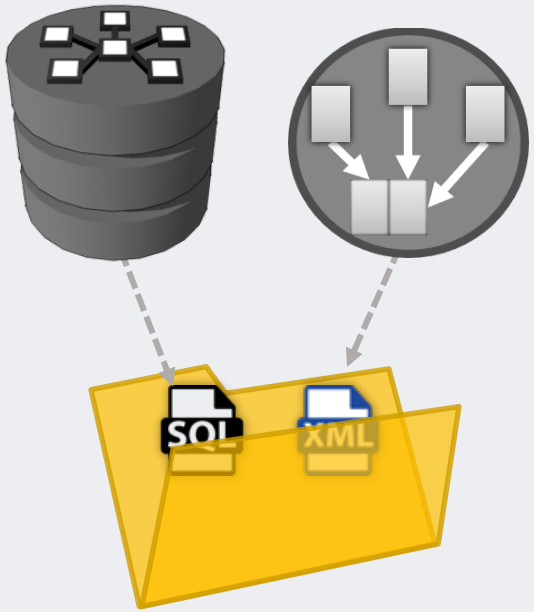
Job Complexity Categorization	
LOW	3341
MEDIUM	840
COMPLEX	311
VERY COMPLEX	58

Transformation Type	# of Occurrences
Aggregator	4
Expression	311
Filter	47
	1
	89
	1
	1
	43
Sequence	37
Source Definition	279
Source Qualifier	232
	272
	4

Function	# of Calls
:UDF.UDFSPACEORNULLTOTILDE	7
DECODE	2
IIF	110
ISNULL	73
LPAD	2
LTRIM	55
MD5	2
RTRIM	55
TO_CHAR	5
TO_DATE	5
TRUNC	1

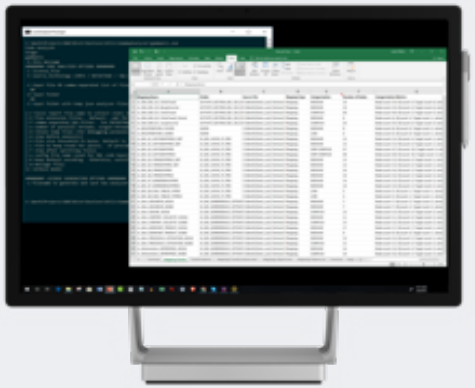
Running Analyzer

Getting empirical counts to size the conversion



Acquire Metadata

Export metadata from Legacy Systems



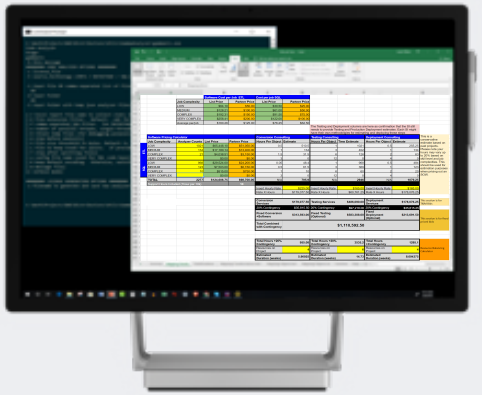
Run Analyzer

Point Analyzer to the folder location where the metadata has been exported

Job Complexity Categorization	
LOW	3341
MEDIUM	840
COMPLEX	311
VERY COMPLEX	58

Acquire Jobs Counts

Copy the complexity counts

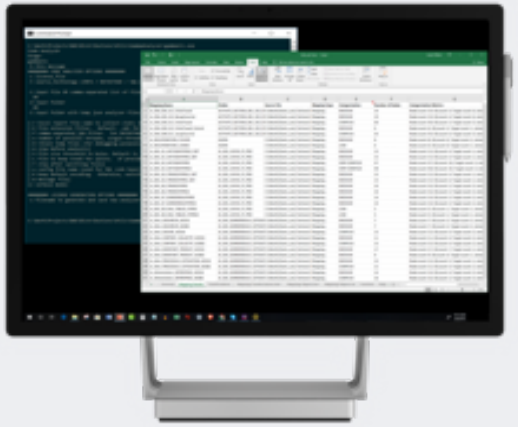


Run Pricing Calculator

Paste the Analyzer complexity counts into the calculator to obtain pricing. Available on partner portal

BladeBridge Onboarding

Becoming a Partner



BladeBridge Agreement

3 page agreement that enables you to have access to the BladeBridge Portal Assets

$5000 Subscription to Analyzer

Enables access to the BladeBridge Analyzer for a year

Onboarding & Enablement

- Analyzer Keys
- Sales/Presales Guidance
- Project Assets
- Demo Jobs

BladeBridge Onboarding

Becoming a Partner

Assist Interpreting Analyzer

Export metadata from Legacy Systems

Jointly Review Pricing Calculator

Review that calculator and that the numbers match Analyzer's output

Evaluate BladeBridge Assistance

BladeBridge provides 1 hour per $10,000 spend on BladeBridge Converter. We recommend buffering that with a few more hours on your first project

Purchase Order for Software

Systems Integrator generates a Purchase Order based on the calculator

Configuration Training

BladeBridge will conduct a training with the Systems Integrator on the Reader, Writer, and Bridge configuration files.

Generate Keys

BladeBridge will generate keys for the converter software based on the Analyzer Checksums

BladeBridge PS Assist

BladeBridge PS team overlap with your project team based on determined duration

DATA+AI SUMMIT 2022

Demo...

Jared Hillam

GTM VP

jhillam@bladebridge.com

Up Next…

Simon Eligulashvili

CoFounder, BladeBridge

DATA+AI SUMMIT 2022

Thank You

Jared Hillam
GTM VP
jhillam@bladebridge.com

Simon Eligulashvili
CoFounder, BladeBridge