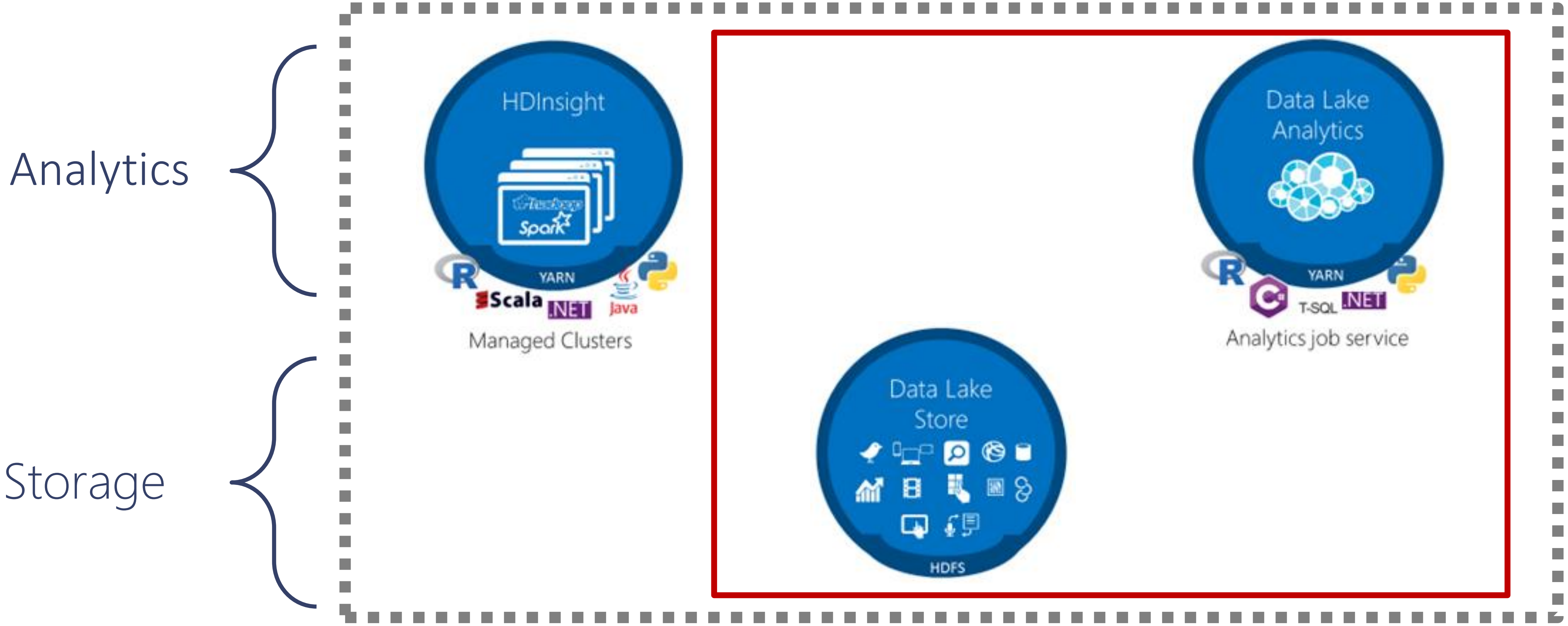
Azure Data Lake AND U-SQL

PLSSUG KATOWICE 2016

PLAN

- Azure Data Lake
  - Azure Data Lake Store
  - Azure Data Lake Analytics
- U-SQL
- Azure Data Lake Live on Azure Demo
- Pricing

Azure Data Lake



Source: Microsoft

Azure Data Lake

AZURE DATA LAKE STORE

- Built for Hadoop
  - WebHDFS-compatible REST interface
- Unlimited storage, petabyte files
- Performance-tuned for big data analytics
- Highly-available and secure
- Integrates with HDInsight, Cloudera, Hortonworks
- Supports files and folders objects
- Files are split apart into Extents (250 MB)
- For availability and reliability, extents are replicated (3 copies).
- Enables: Parallel read and Parallel write

AZURE DATA LAKE ANALYTICS

A distributed analytics service built on Apache YARN that dynamically scales to your needs

- Pay **PER QUERY** & Scale **PER QUERY**
- **FEDERATED QUERY** across Azure data sources
- Includes **U-SQL**, a language that unifies the benefits of SQL with the expressive power of C#
- No limits to **SCALE**
- Optimized to work with **ADL STORE**

Source: Microsoft

Why new language for Big Data

TRADITIONAL SQL FOR BIG DATA:

+ Declarative

- Hard to extend

TRADITIONAL PROGRAMMING LANGUAGES FOR BIG DATA:

+ Extensible

- Requires a lot of code/knowledge to scale and perform

DECLARATIVITY

+

EXTENSIBILITY =

U-SQL

Source: M. Rys

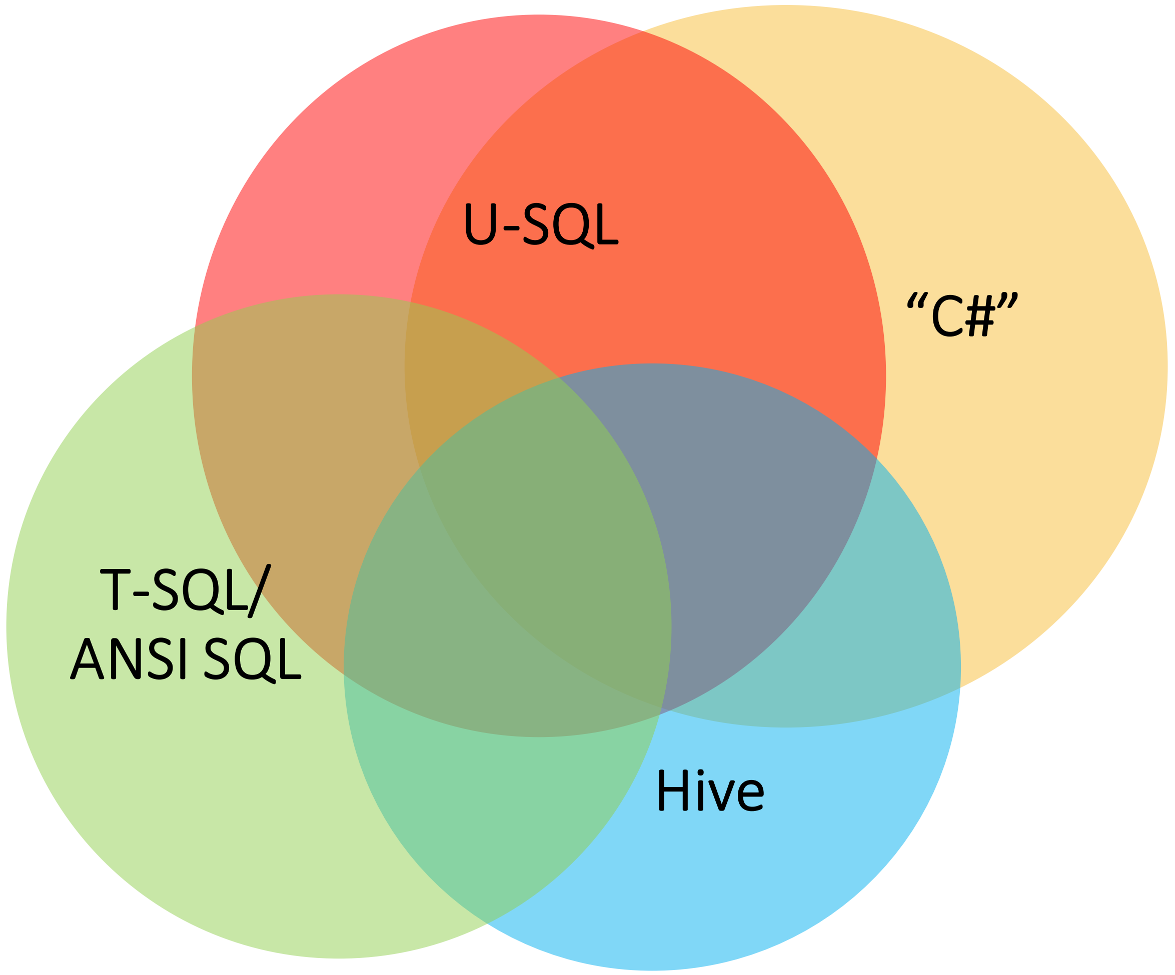
U –SQL
A new language for Big Data

Familiar syntax to millions of SQL & .NET developers

Unifies declarative nature of SQL with the imperative power of C#

Unifies structured, semi-structured and unstructured data

Distributed query support over all data



Source: M. Rys

U-SQL SCRIPT/JOB

```
DECLARE @inputPostCodes string =
@"mySamples/UK/ukpostcodes.csv";

@postCodes =
    EXTRACT id string,
            postcode string,
            latitude string,
            longitude string
    FROM @inputPostCodes
    USING Extractors.Csv(skipFirstNRows : 1);

OUTPUT @postCodes
TO @"ukpostcodes.txt"
USING Outputters.Text();
```

DECLARE (Optional)

EXTRACT (or SELECT)

Apply Schema on read

OUTPUT (or INSERT)

CASE SENSITIVE

(“KEYWORDS” UPPER CASE)

U-SQL DATA TYPES

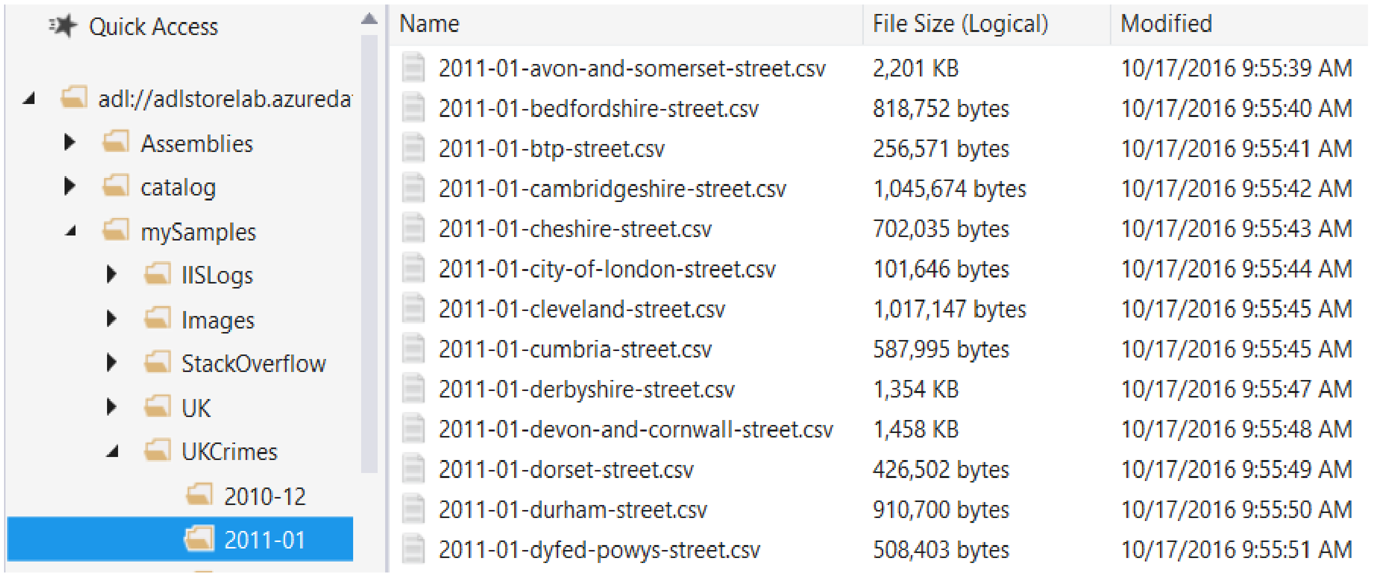
```
DECLARE @text1 string = “PLSSUG KATOWICE";
DECLARE @text4 string = "BEGIN" + @text1 + "END";
DECLARE @text6 string =
string.Join(" ", new String[]{@text1, “2016"});

DECLARE @numeric1 sbyte = 0;
DECLARE @numeric2 short = 1;
DECLARE @numeric3 int = 2;
DECLARE @numeric4 long = 3L;
DECLARE @numeric5 float = 4.0f;
DECLARE @numeric6 double = 5.0;

DECLARE @d1 DateTime = System.DateTime.Parse("1979/03/31");
DECLARE @d2 DateTime = DateTime.Now;

DECLARE @misc1 bool = true;
DECLARE @misc2 Guid =
System.Guid.Parse("BEF7A4E8-F583-4804-9711-7E608215EBA6");

DECLARE @misc4 byte [] = new byte[] { 0, 1, 2, 3, 4};

SQL.ARRAY<T> == IList<T> EXPLODE
SQL.MAP<T,U> == IDictionary<T,U> EXPLODE

@DataSets    == TABLE
```

```
 @m = SELECT new SqlArray<string>
 (
            tweet.Split(
            new char[]{' '}).Where(x => x.StartsWith("@")))  AS
mentions
      FROM @t;

 @m = SELECT m.Substring(1) AS m
           , "mention" AS category
      FROM @m CROSS APPLY EXPLODE(mentions) AS t(m)


 @ds =
     SELECT content,fileName, new SQL.MAP<int,string>() AS
 colors
     FROM @rs;

 @ds =
     PROCESS @ds
     PRODUCE content,colors,fileName
             READONLY fileName
     USING new ImageColorsProcessor(4);

 @ds =
     SELECT fileName,
            order,
            colorName
     FROM @ds
          CROSS APPLY
              EXPLODE(colors) AS colors(order, colorName);
```

U-SQL EXTRACTORS, OUTPUTTERS and FILESETS

- “EMBEDDED” EXTRACTORS AND OUTPUTTERS
  - Csv, Text ,Tsv
  - API IExtractor , Ioutputter
- FILESETS

```
DECLARE @inputCrimes = @"mySamples/UKCrimes/{Date:yyyy}-
{Date:MM}/{Input}-street.csv";
@crimes =
    EXTRACT CrimeID string,
            Month string,
            ReportedBy string,
            FallsWithin string,
            Longitude string,
            Latitude string,
            Location string,
            LSOACode string,
            LSOAName string,
            CrimeType string,
            LastOutcomeCategory string,
            Context string,
            Date DateTime,
            Input string
    FROM @inputCrimes
    USING Extractors.Csv(silent :
false,skipFirstNRows:1);
```



Name	File Size (Logical)	Modified
2011-01-avon-and-somerset-street.csv	2,201 KB	10/17/2016 9:55:39 AM
2011-01-bedfordshire-street.csv	818,752 bytes	10/17/2016 9:55:40 AM
2011-01-btp-street.csv	256,571 bytes	10/17/2016 9:55:41 AM
2011-01-cambridgeshire-street.csv	1,045,674 bytes	10/17/2016 9:55:42 AM
2011-01-cheshire-street.csv	702,035 bytes	10/17/2016 9:55:43 AM
2011-01-city-of-london-street.csv	101,646 bytes	10/17/2016 9:55:44 AM
2011-01-cleveland-street.csv	1,017,147 bytes	10/17/2016 9:55:45 AM
2011-01-cumbria-street.csv	587,995 bytes	10/17/2016 9:55:45 AM
2011-01-derbyshire-street.csv	1,354 KB	10/17/2016 9:55:47 AM
2011-01-devon-and-cornwall-street.csv	1,458 KB	10/17/2016 9:55:48 AM
2011-01-dorset-street.csv	426,502 bytes	10/17/2016 9:55:49 AM
2011-01-durham-street.csv	910,700 bytes	10/17/2016 9:55:50 AM
2011-01-dyfed-powys-street.csv	508,403 bytes	10/17/2016 9:55:51 AM

U-SQL FILTERING AND SORTING

- WHERE
  - AND & OR
  - ==, >=, != (C# OPERATOR(s))
  - CONTAINS (C# string)
- ORDER BY
  - ROWSETS
    - requires a FETCH FIRST
  - OUTPUTS

```
@distances =
    SELECT CrimeId,
           CityName,
           CrimeType,
           Year,
           Month
    FROM @merged
    WHERE CrimeType.StartWith("Soc")
AND Year == 2016
ORDER BY Month DESC
FETCH FIRST 10 ROWS;
```

U -SQL – AGGREGATIONS AND WINDOW FUNCTIONS

- **GROUP BY**
- **HAVING**
- **AGGREGATIONS**
  - MAX ,MIN, SUM, COUNT …
  - ARRAY_AGG
- **RANKING FUNCTIONS**
  - RANK, DENSE_RANK, NTILE, ROW_NUMBER
- **ANALIYTIC WINDOW FUNCTIONS**
  - CUME_DIST, PERCENT_RANK, PERCENTILE_CONT, PERCENTILE_DISC,CUME_DIST

```
@output =
     SELECT
         MAX(Duration) AS DurationMax,
         MIN(Duration) AS DurationMin
 FROM @searchlog
     GROUP BY Region
     HAVING DurationMin > 1;
 @result =
 SELECT
     *,
     ROW_NUMBER() OVER
     (PARTITION BY Vertical ORDER BY Latency) AS RowNumber,
     RANK() OVER (PARTITION BY Vertical ORDER BY Latency) AS Rank,
     DENSE_RANK()
      OVER (PARTITION BY Vertical ORDER BY Latency) AS DenseRank
 FROM @querylog;
```

U-SQL JOINS

- INNER JOIN
- FULL OUTER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- CROSS JOIN
- LEFT SEMIJOIN (IN)
- RIGHT SEMIJOIN (IN)
- LEFT ANTISEMIJOIN (NOT IN)
- RIGHT ANTISEMIJOIN (NOT IN)

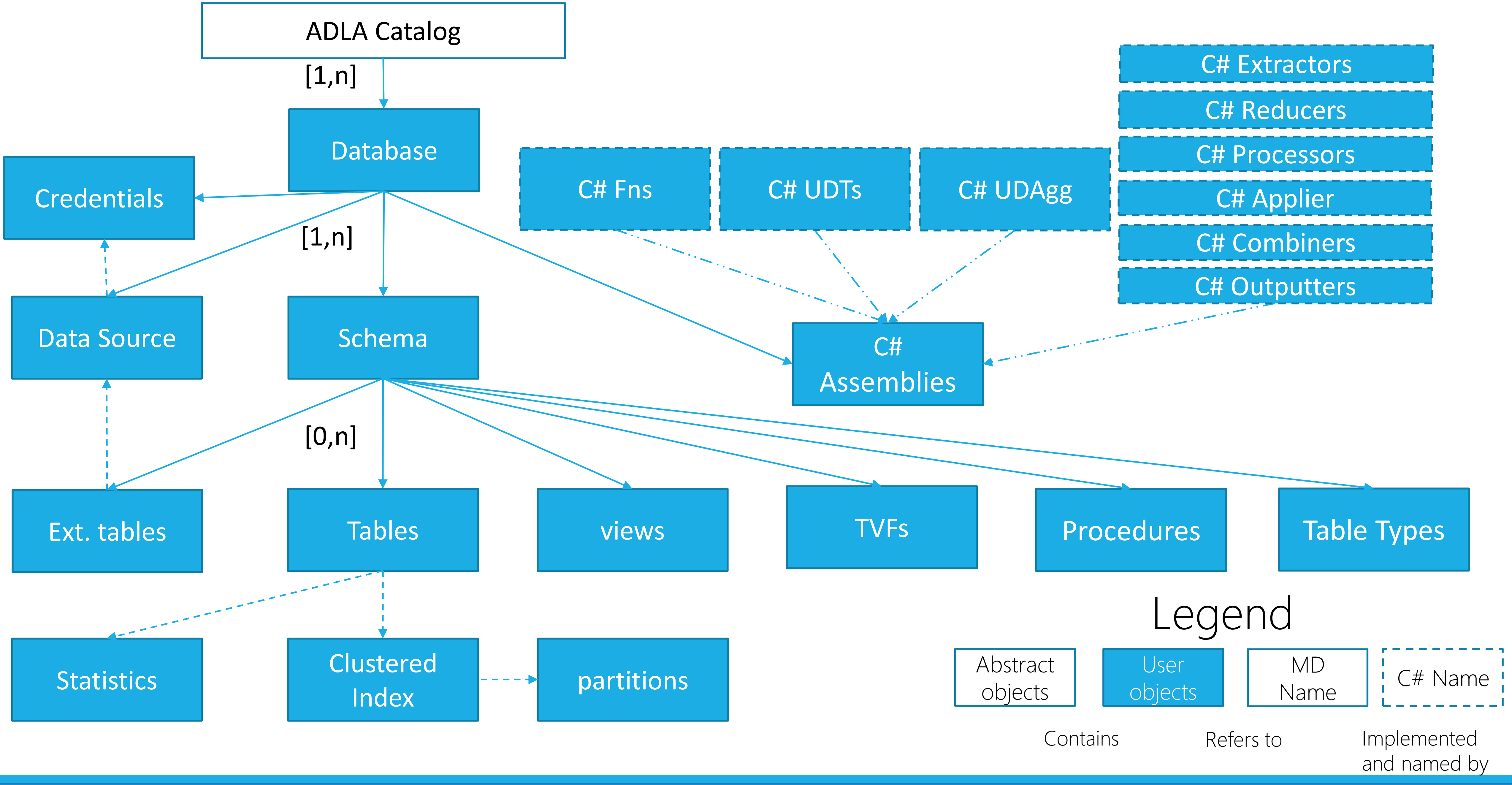
```
@topCitiesWithGPS =
SELECT
tc.name,tc.population,
pc.latitude,pc.longitude
FROM @topCities AS tc
JOIN @postCodes AS pc
            ON pc.postcode ==
tc.postcode;
```

U-SQL

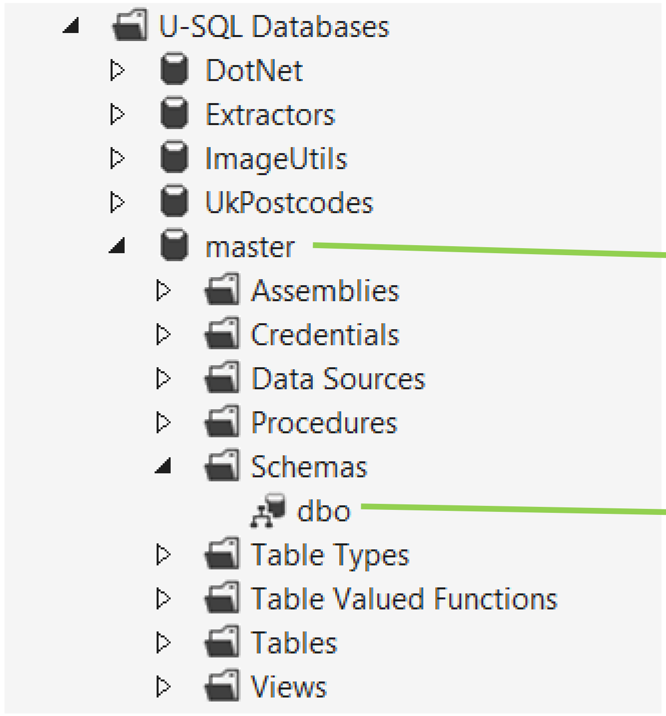
Let’s show the U-SQL

DEMO

(Visual Studio and U-SQL Project)


ADLA Catalog
[1,n]
Database
Credentials
[1,n]
Data Source
Schema
C# Fns
C# UDTs
C# UDAgg
C# Extractors
C# Reducers
C# Processors
C# Applier
C# Combiners
C# Outputters
C# Assemblies
[0,n]
Ext. tables
Tables
views
TVFs
Procedures
Table Types
Statistics
Clustered Index
partitions
Legend
Abstract objects
User objects
MD Name
C# Name
Contains
Refers to
Implemented and named by

U-SQL DATABASES AND SCHEMES



```
CREATE DATABASE IF NOT EXISTS UKCrimes;
USE DATABASE UKCrimes;
CREATE SCHEMA IF NOT EXISTS cr;
```

Default database

Default schema

U-SQL TABLES

- MANAGED TABLES and EXTERNAL TABLES
- ONLY INSERT
- CONSISTS OF FOUR THINGS:
  - A NAME
  - COLUMNS
  - A CLUSTERED INDEX
  - PARTITIONING SCHEME

```
DROP TABLE IF EXISTS vehiclesP;
CREATE TABLE vehiclesP(
      vehicle_id int
    , entry_id long
    , event_date DateTime
    , latitude float
    , longitude float
    , speed int
    , direction string
    , trip_id int?
    , INDEX idx CLUSTERED (vehicle_id ASC)
      PARTITIONED BY (event_date)
      DISTRIBUTED BY HASH (vehicle_id) INTO 4
);
```

U-SQL VIEWS and FUNCTIONS

VIEWS

```
CREATE VIEW IF NOT EXISTS vCrimes
    AS
EXTRACT CrimeID string,
        Month string,
        Date DateTime,
        Input string
FROM @"\UKCrimesCities\{Date:yyyy}-{Date:MM}\{Input}-
street.csv"
USING Extractors.Csv(silent : false,
 skipFirstNRows : 1);
```

FUNCTIONS (TVF)

```
CREATE FUNCTION tvf_Crimes(@input string)
RETURNS @result TABLE(CrimeID string,
 Month string)
AS
BEGIN
    @crimes =
    EXTRACT CrimeID string,
            Month string
    FROM @input
    USING Extractors.Csv(silent : false,
skipFirstNRows:1);

    @result = SELECT CrimeID,
            Month
            Input FROM @crimes;
    END;
```

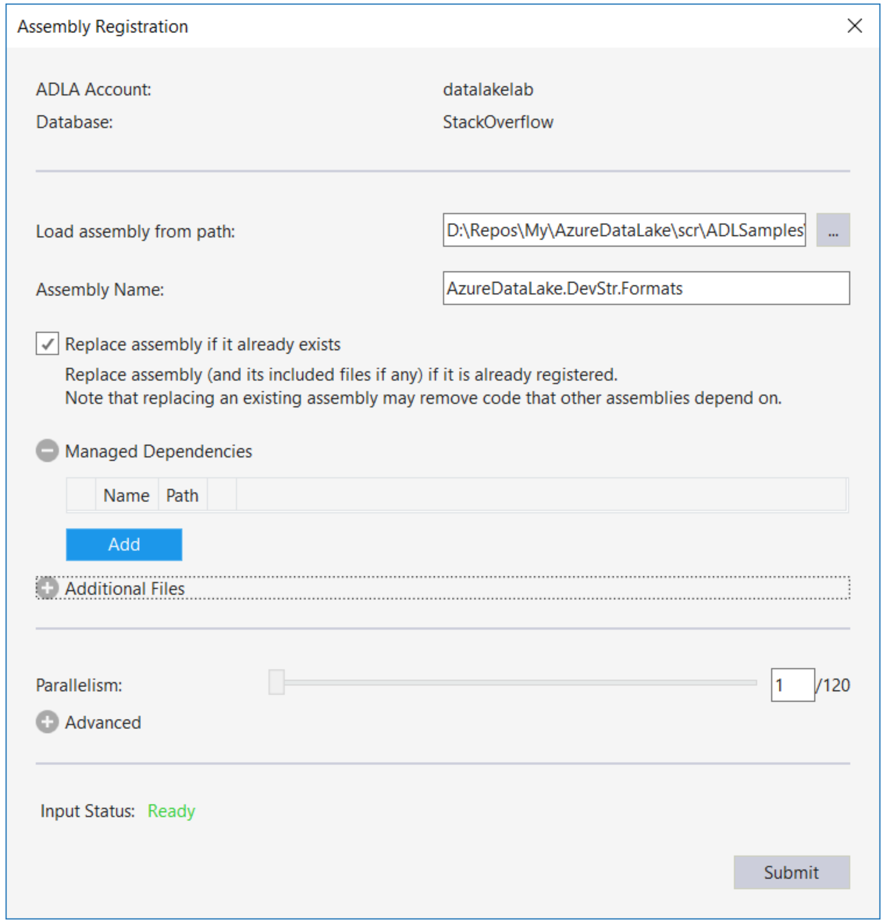
U-SQL C# METHODS

```
@distances =
    SELECT CrimeId,
           CityName,
           CrimeType,
           Year,
           Month,
           Gps.ComputeDistance
           (sLatitude, sLongitude, dLatitude, dLongitude)
AS Distance
    FROM @merged;
```

```
public static double ComputeDistance(double sLat, double
sLong, double dLat, double dLong)
        {
            var locA = new GeoCoordinate(sLat, sLong);
            var locB = new GeoCoordinate(dLat, dLong);
            return locA.GetDistanceTo(locB); // metres
        }
```

C# Method

U-SQL USING ASSEMBLIES



Assembly Registration

ADLA Account: datalakelab
Database: StackOverflow

Load assembly from path: D:\Repos\My\AzureDataLake\scr\ADLSamples

Assembly Name: AzureDataLake.DevStr.Formats

[✓] Replace assembly if it already exists
Replace assembly (and its included files if any) if it is already registered.
Note that replacing an existing assembly may remove code that other assemblies depend on.

Managed Dependencies

Name	Path

Parallelism: 1 /120

Input Status: Ready

```
DECLARE @myAssemblyPath string =
@"D:\Repos\AzureDataLake.DevStr.Formats\bin\Debug\";

DECLARE @myAssemblyName string =
@myAssemblyPath+"AzureDataLake.DevStr.Formats.dll";

CREATE DATABASE IF NOT EXISTS Extractors;

USE DATABASE Extractors;

DROP ASSEMBLY IF EXISTS MyExtractors;

CREATE ASSEMBLY MyExtractors

FROM @myAssemblyName;
```

U-SQL USING ASSEMBLIES

```
DECLARE @imgFile string = @"D:\Help\BIGDATA\Images\{fileName}.jpg";
USE DATABASE Extractors;
REFERENCE ASSEMBLY MyExtractors;
USING BinaryExtractor = AzureDataLake.DevStr.Formats.BinaryContentExtractor;
REFERENCE ASSEMBLY ImageUtils.ImageUtils;
USING ImageColorsProcessor = AzureDataLake.DevStr.ImageUtils.ImageColorProducer;
@rs =
    EXTRACT content byte[],
            fileName string
    FROM @imgFile
    USING new BinaryExtractor();

@ds =
    SELECT content,fileName, new SQL.MAP<int,string>() AS colors
    FROM @rs;

@ds =
    PROCESS @ds
    PRODUCE content,colors,fileName
            READONLY fileName
    USING new ImageColorsProcessor(4);
```

Reference

Alias

External Extractor

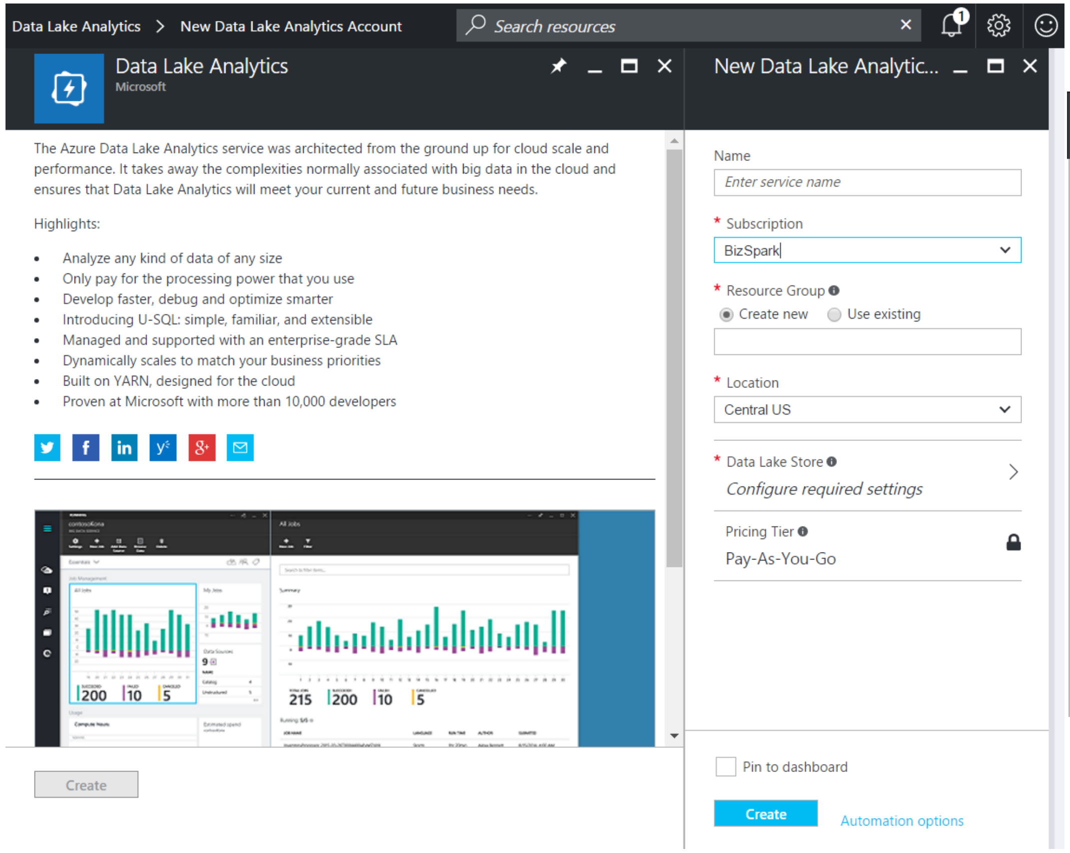
External Processor

U-SQL

DEMO

(Assemblies)

Azure Data Lake Analytics



Data Lake Analytics
Microsoft

The Azure Data Lake Analytics service was architected from the ground up for cloud scale and performance. It takes away the complexities normally associated with big data in the cloud and ensures that Data Lake Analytics will meet your current and future business needs.

Highlights:

- Analyze any kind of data of any size
- Only pay for the processing power that you use
- Develop faster, debug and optimize smarter
- Introducing U-SQL: simple, familiar, and extensible
- Managed and supported with an enterprise-grade SLA
- Dynamically scales to match your business priorities
- Built on YARN, designed for the cloud
- Proven at Microsoft with more than 10,000 developers

New Data Lake Analytic...

Name: Enter service name

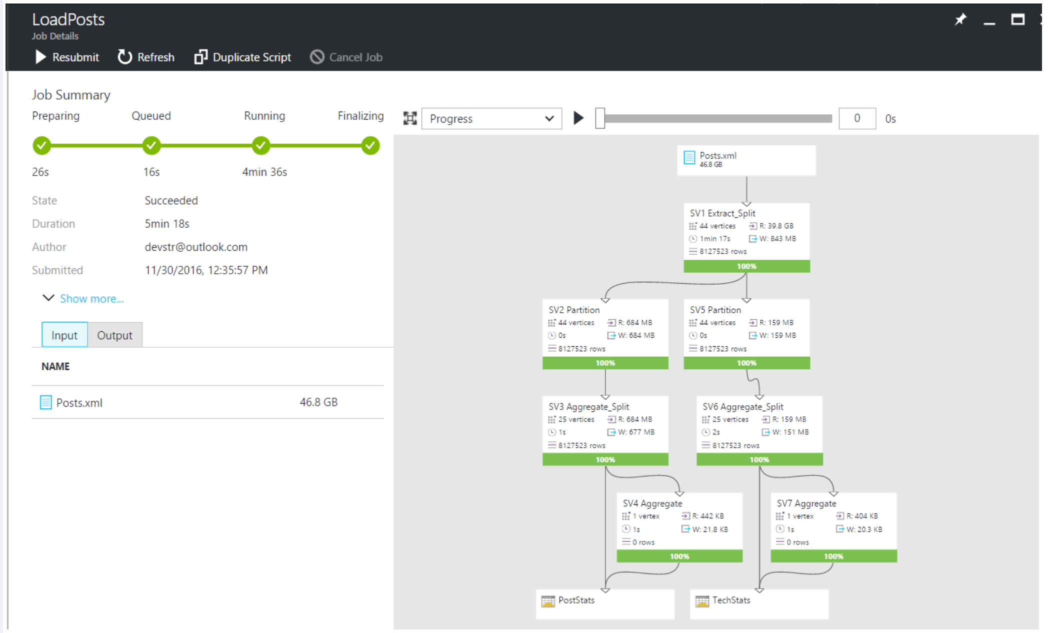
* Subscription: BizSpark

* Resource Group: [✓ Create new] Use existing

* Location: Central US

* Data Lake Store: Configure required settings

Pricing Tier: Pay-As-You-Go



LoadPosts
Job Details

Job Summary

Preparing	Queued	Running	Finalizing
26s	16s	4min 36s	

State	Succeeded
Duration	5min 18s
Author	devstr@outlook.com
Submitted	11/30/2016, 12:35:57 PM

Show more...

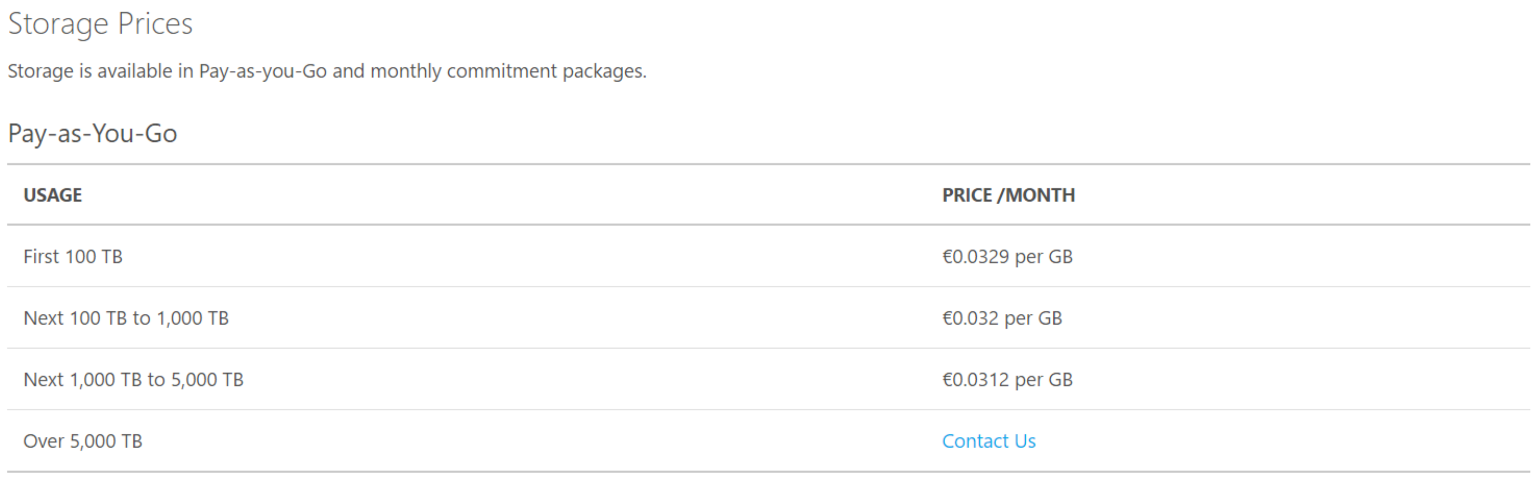
Input | Output

NAME	
Posts.xml	46.8 GB

Azure Data Lake Analytics

AZURE DEMO
(TechRadar, UKCrimes)

Azure Data Lake Store Pricing

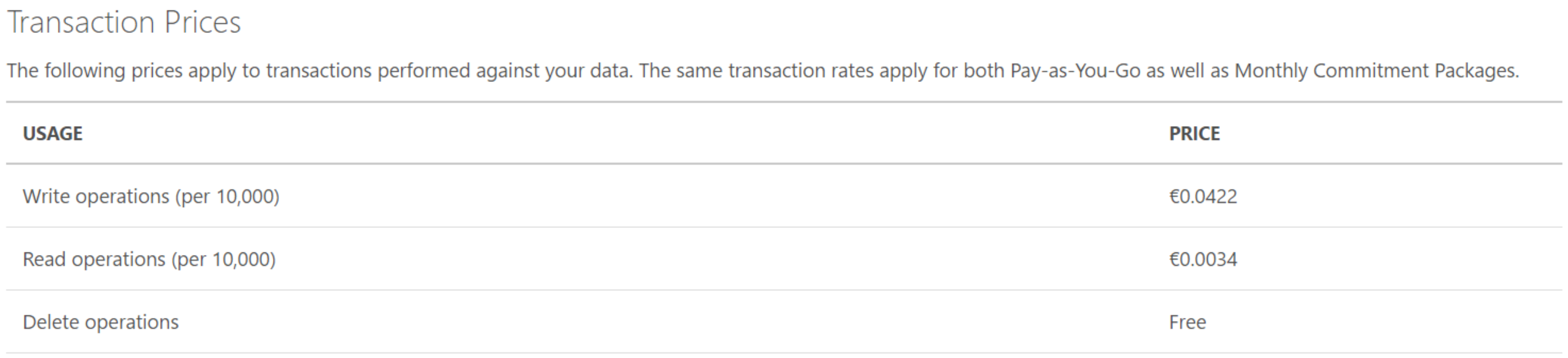


Storage Prices

Storage is available in Pay-as-you-Go and monthly commitment packages.

Pay-as-You-Go

USAGE	PRICE /MONTH
First 100 TB	€0.0329 per GB
Next 100 TB to 1,000 TB	€0.032 per GB
Next 1,000 TB to 5,000 TB	€0.0312 per GB
Over 5,000 TB	Contact Us

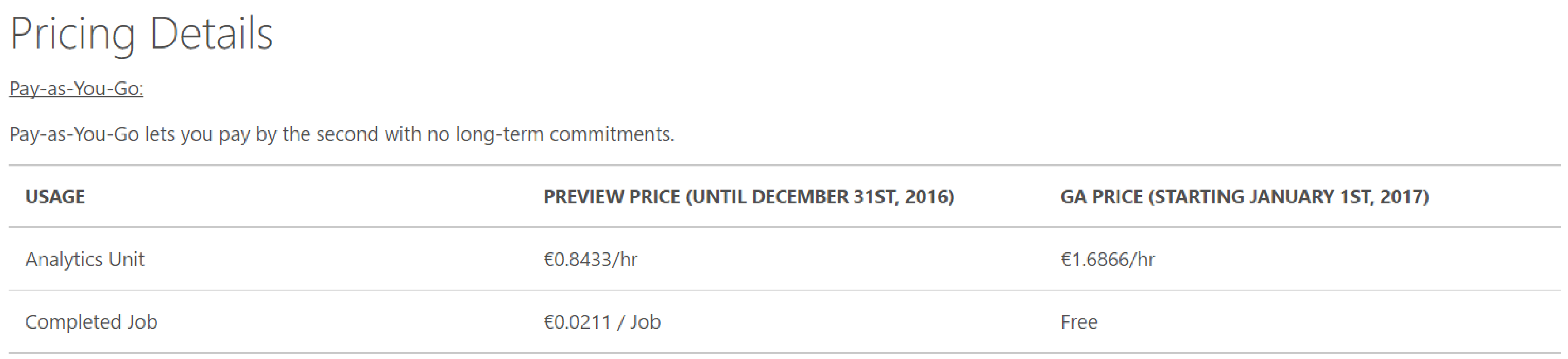


Transaction Prices

The following prices apply to transactions performed against your data. The same transaction rates apply for both Pay-as-You-Go as well as Monthly Commitment Packages.

USAGE	PRICE
Write operations (per 10,000)	€0.0422
Read operations (per 10,000)	€0.0034
Delete operations	Free

Data Lake Analytics Pricing



Pricing Details

Pay-as-You-Go:

Pay-as-You-Go lets you pay by the second with no long-term commitments.

USAGE	PREVIEW PRICE (UNTIL DECEMBER 31ST, 2016)	GA PRICE (STARTING JANUARY 1ST, 2017)
Analytics Unit	€0.8433/hr	€1.6866/hr
Completed Job	€0.0211 / Job	Free

$$JobCost = (seconds \times ADLU \times ADLU\ Cost)\ /3600 + \textit{Completed Job Cost} + \textit{Data Lake Transactions Costs}$$

Azure Data Lake Analytics Unit

Parallelism N = N ADLAUs

1 ADLAU ~=

- A VM with 2 cores and 6 GB of memory

Azure Data Lake Analytics

AZURE DEMO

(AU Usage Modeler)

Azure Data Lake

Examples:

- https://github.com/devstr/usql

Contact:

tomasz.k.krawczyk@gmail.com