Technology Overview

November 2023

Bob Foreman
Software Engineering Lead
LexisNexis Risk Solutions

HPCC Systems:
End to End Data Lake Management

Completely free

open-source data lake solution

Out of the box capabilities for consistency and ease of use

Less coding

and more using (even though we love to code)

We are your one stop shop for all your data integration, querying and analytical needs

HPCC SYSTEMS

Why does HPCC Systems exist?

- ✓ HPCC defined is a *distributed data and parallel processing* platform.
- ✓ It was NOT developed with the idea of selling the technology to anybody else!
- ✓ It was all created only to solve some of the data-handling problems that we encountered as we were developing our products.

HPCC Systems Evolution

2001

Original version of HPCC Systems released

2011

Open source Apache license and code release to GitHub

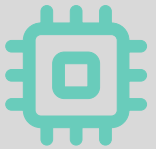
Exceeded market-leading performance benchmark achieved

2012 – 16

Continuous **QUALITY-FOCUSED** improvements

Better support and training with improved integration — faster and easier to use

2017-2023



Improved processing architecture

IoT enabled

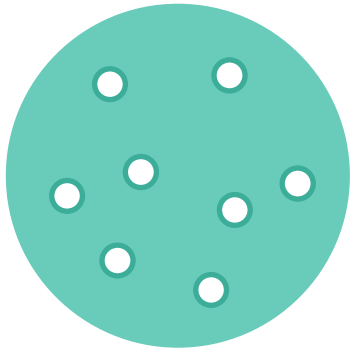
ML Expansion!

Cloud Native!

HPCC SYSTEMS

The Data Centric Approach

A single source of data is insufficient to overcome inaccuracies

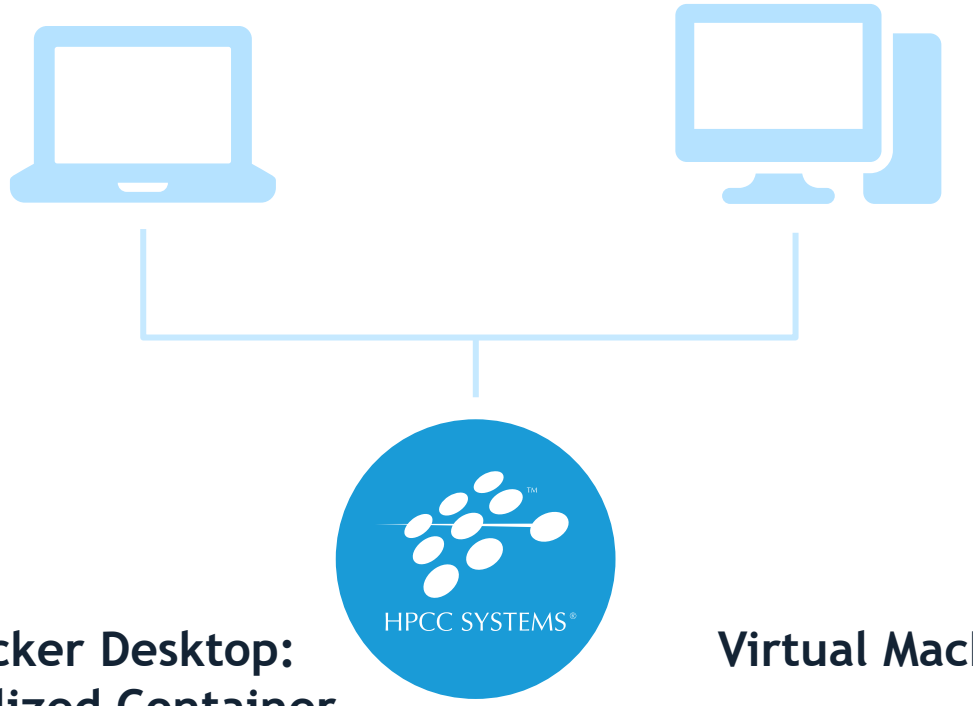


Our platform is built on the premise of absorbing data from **many data sources** and transforming them to **actionable smart data**

HPCC SYSTEMS

Scale from Small to Big

The stack can run on a single laptop or desktop.



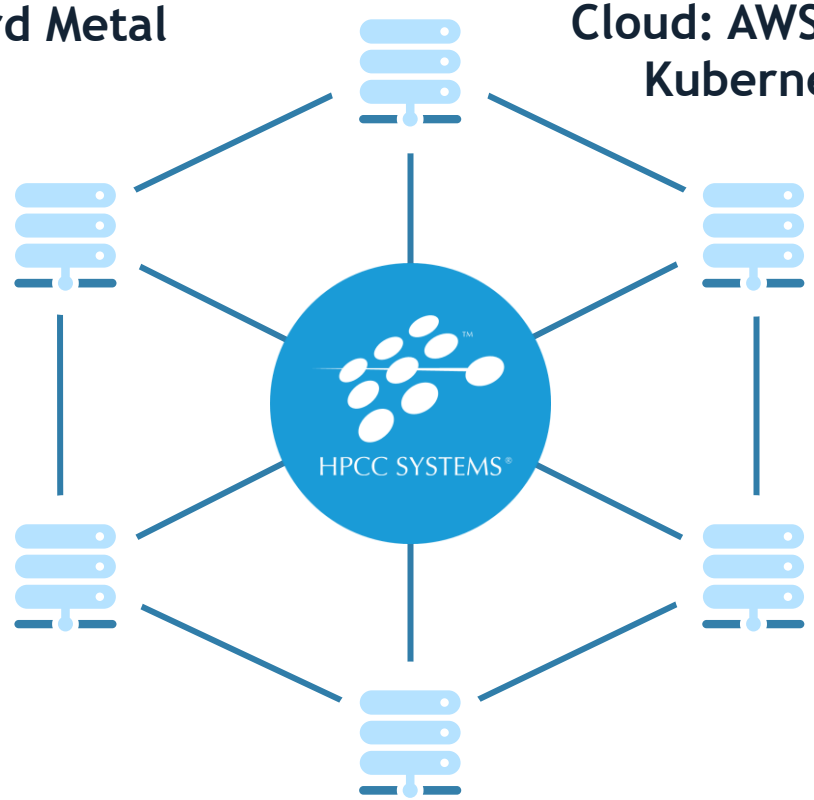
Docker Desktop: Localized Container

Virtual Machine

In more sophisticated cases, HPCC Systems run *clusters*, hundreds of servers working as a single processing entity, to transform and deliver big data.

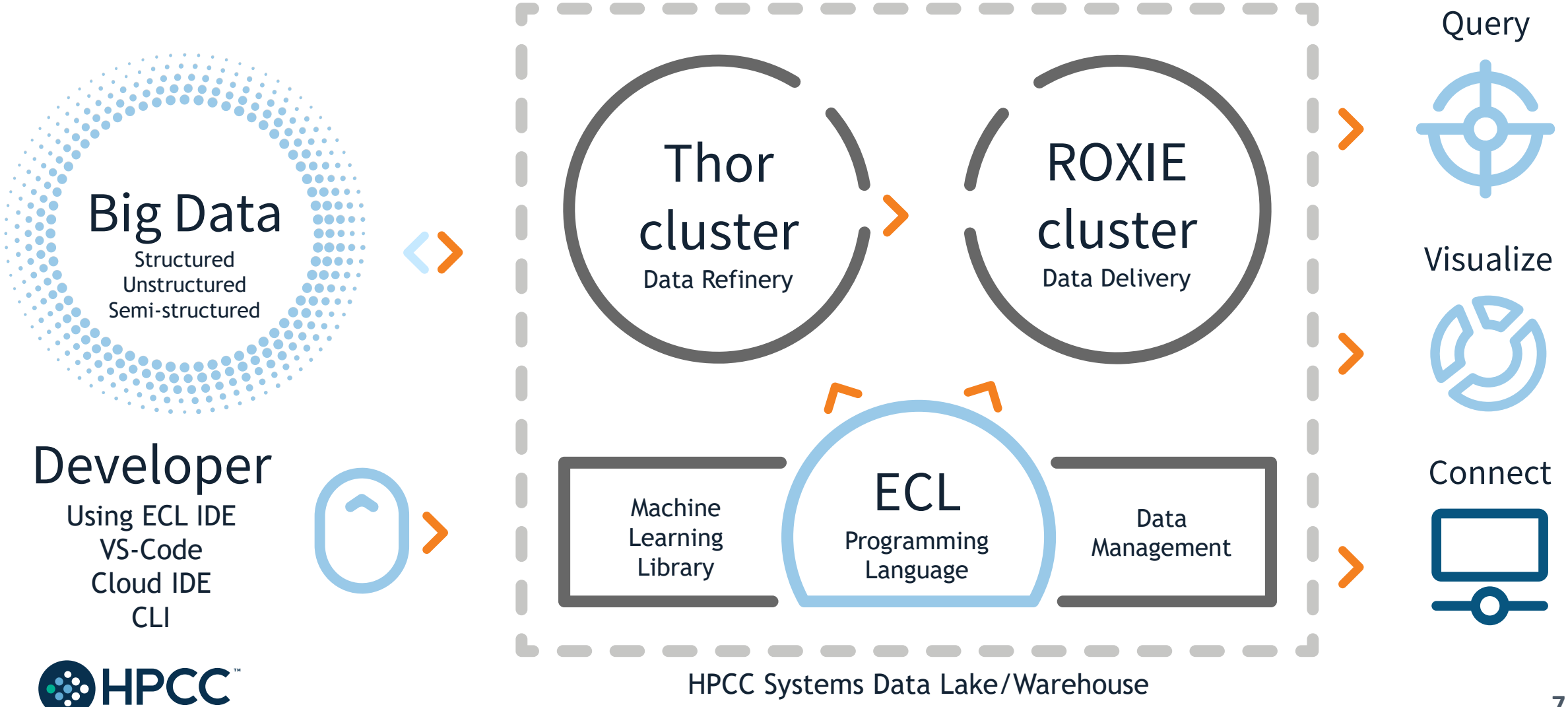
Hard Metal

Cloud: AWS/Azure Kubernetes

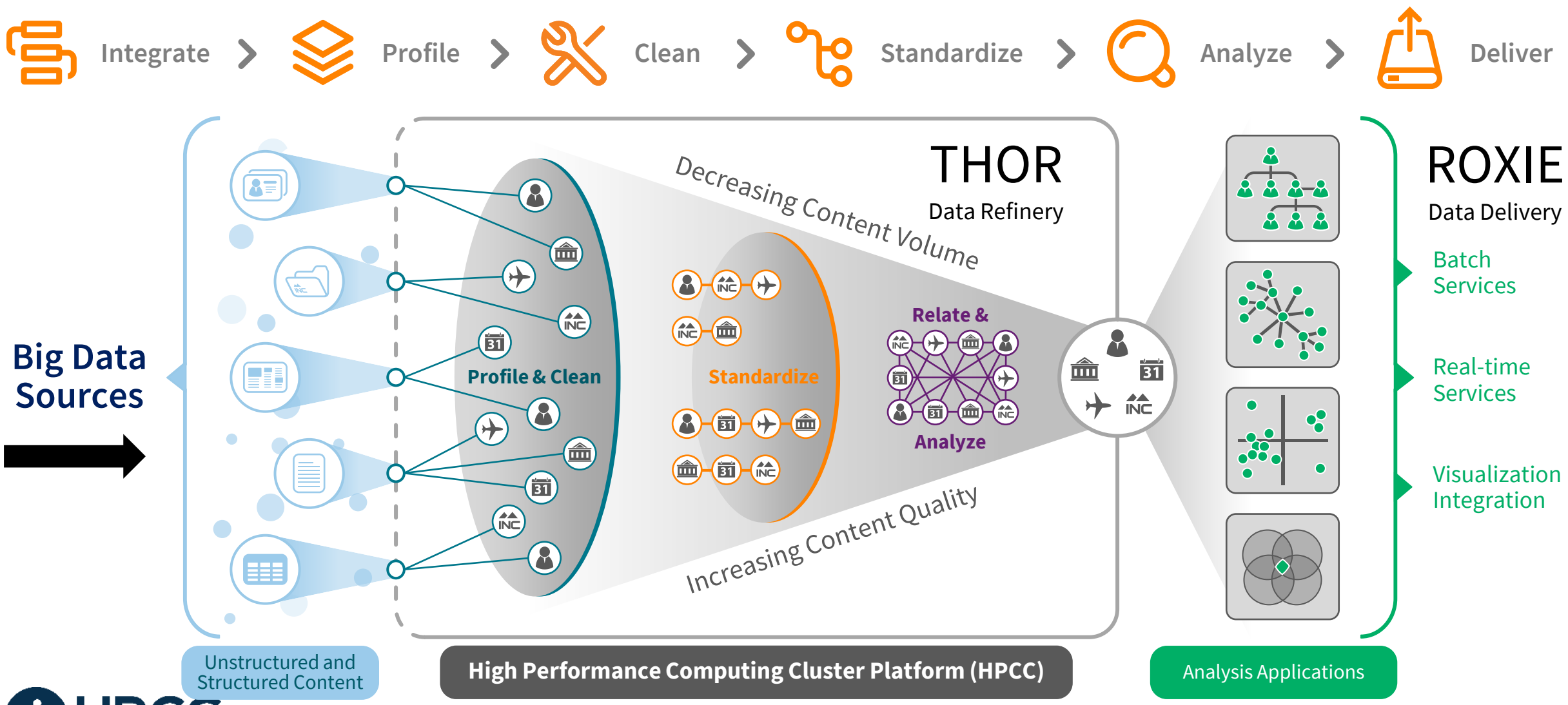


HPCC SYSTEMS

The HPCC Systems Components



HPCC Systems (Small to Big Data) ETL



HPCC SYSTEMS

Technology — The Open-Source Stack

Thor: Data Refinery Cluster

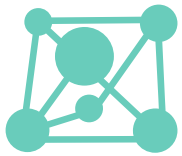
Extraction, loading, cleansing, transforming, linking and indexing

ROXIE: Data Delivery Engine

Rapid data delivery cluster with high-performance online query delivery for big data

Data Management Tools

Data profiling, cleansing, snapshot data updates, consolidation, job scheduling and automation



Machine Learning Library

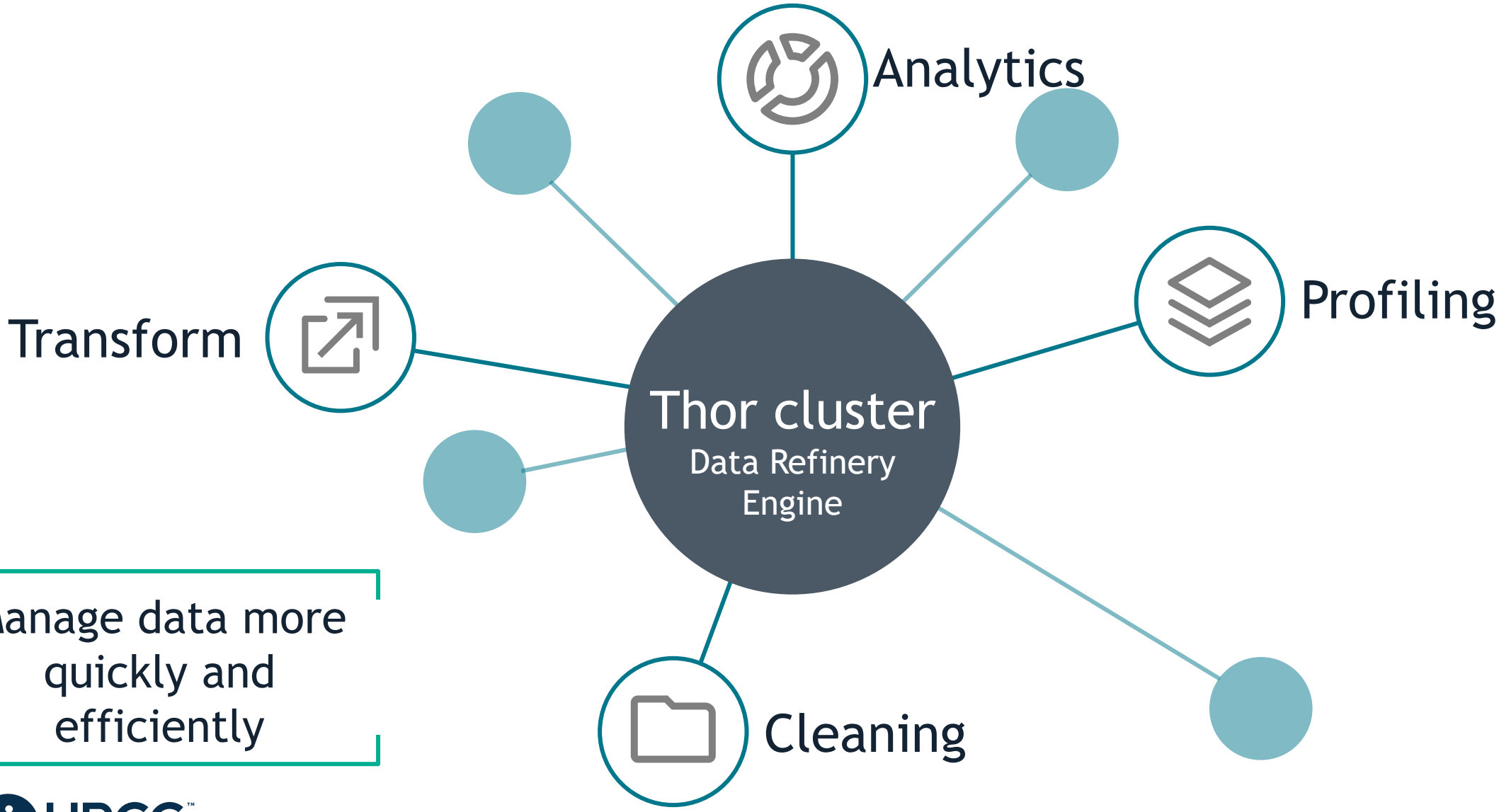
Linear regression, logistic regression, decision trees and random forests

Connectivity & Third-Party Tools

New plugins to help integrate third party tools with the HPCC Systems platform

HPCC SYSTEMS

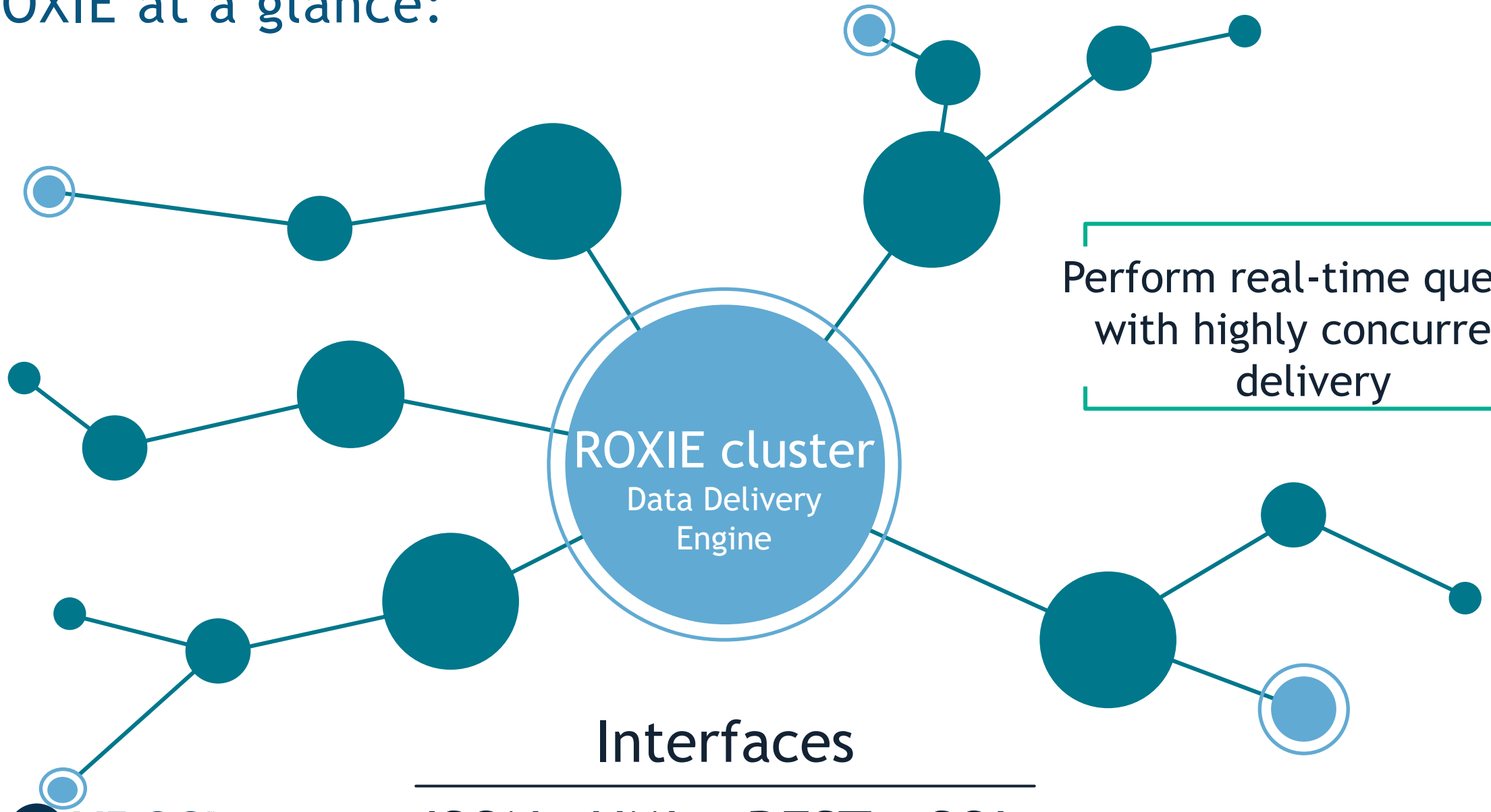
THOR at a glance:



Manage data more quickly and efficiently

HPCC SYSTEMS

ROXIE at a glance:



Perform real-time queries with highly concurrent delivery

Interfaces

JSON XML REST SQL

HPCC SYSTEMS

An Introduction to ECL

ECL
Enterprise Control Language

```
IMPORT $, STD, ML;
EXPORT Func(UNSIGNED C, UNSIGNED2 Dist, UNSIGNED size, STRING Fld, REAL Parm1=0, REAL Parm2=0, REAL Parm3=0) := MODULE
  SHARED Node := STD.system.Thorlib.Node()+1;
  SHARED PersistPrefix := $.Parms.PersistPrefix;
  SHARED TotalRecs := $.Parms.RecCnt*CLUSTERSIZE;
  SHARED UIDval := IF(C=1, node, node + ((C-1)*CLUSTERSIZE));
  SHARED BOOLEAN IsRandFile := $.Parms.Randomness = $.ut.RandomSrc.file;
  SHARED Normal := FUNCTION
    Thisdist := IF(Parm3=0,
                   ML.Distribution.Normal(Parm1, Parm2),
                   ML.Distribution.Normal(Parm1, Parm2, Parm3));
    RetVals := ML.Distribution.GenData(TotalRecs, Thisdist, 1) : PERSIST(PersistPrefix + 'NormalDistInt' + Fld,EXPIRE(1));
    RETURN RetVals;
    END;
  SHARED Normal2 := FUNCTION
    Thisdist := IF(Parm3=0,
                   ML.Distribution.Normal2(Parm1, Parm2),
                   ML.Distribution.Normal2(Parm1, Parm2, Parm3));
    RetVals := ML.Distribution.GenData(TotalRecs, Thisdist, 1) : PERSIST(PersistPrefix + 'Normal2DistInt' + Fld,EXPIRE(1));
    RETURN RetVals;
    END;
  SHARED Uniform := FUNCTION
    Thisdist := IF(Parm3=0,
                   ML.Distribution.Uniform(Parm1, Parm2),
                   ML.Distribution.Uniform(Parm1, Parm2, Parm3));
    RetVals := ML.Distribution.GenData(TotalRecs, Thisdist, 1) : PERSIST(PersistPrefix + 'UniformDistInt' + Fld,EXPIRE(1));
    RETURN RetVals;
    END;
  SHARED StudentT := FUNCTION
    Thisdist := ML.Distribution.StudentT(Parm1, Parm2);
    RetVals := ML.Distribution.GenData(TotalRecs, Thisdist, 1) : PERSIST(PersistPrefix + 'StudentTDistInt' + Fld,EXPIRE(1))
    RETURN RetVals;
```

- Transparent and implicitly parallel programming language
- Both powerful and flexible

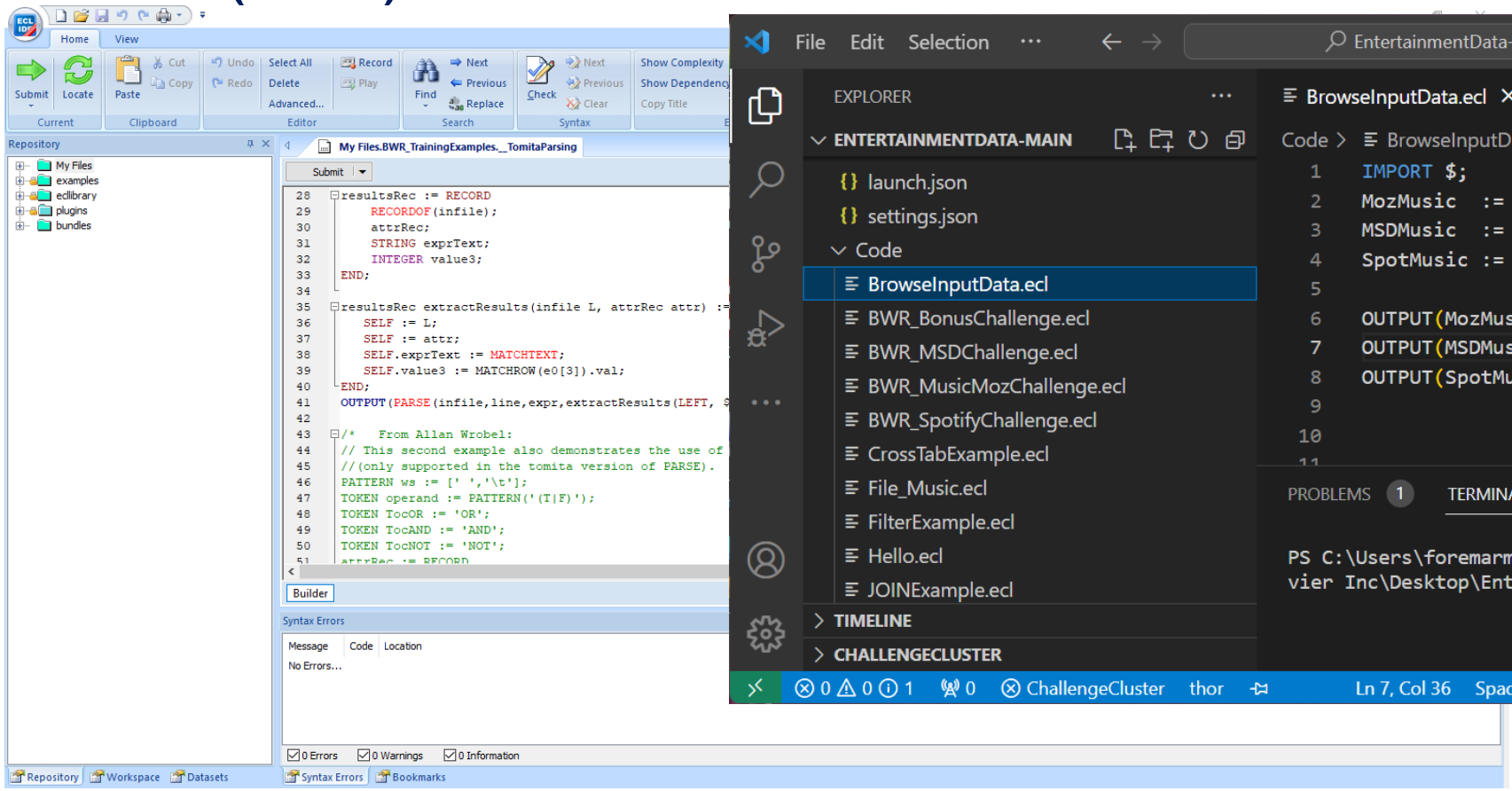
- Optimized for data-intensive operations, declarative, non-procedural and dataflow oriented
- Uses intuitive syntax which is modular, reusable, extensible and highly productive

How to do it vs. What to do

HPCC SYSTEMS

Integrated Development Environments

ECL IDE (Win)



Visual Studio Code (Ux/MacOS)

And CLI too! ECL.EXE

Integrated Development Environments (IDEs)

HPCC Cluster ECL Watch:

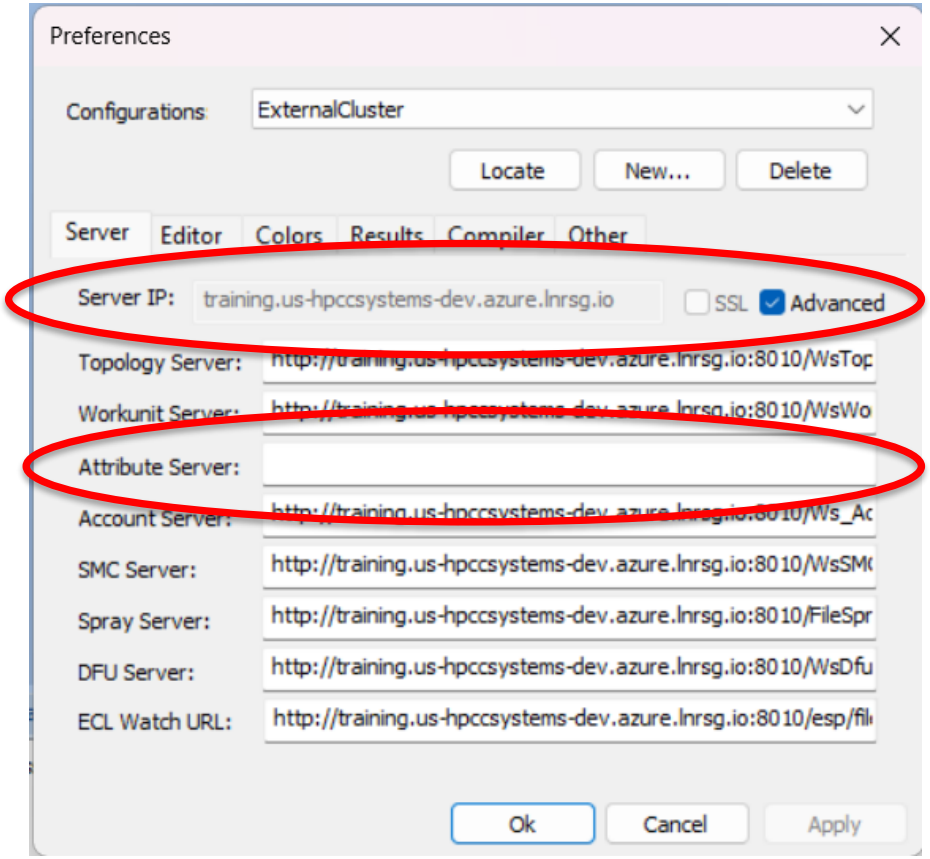
http://training.us-hpccsystems-dev.azure.lnrsg.io:8010/

VS-Code

```
"configurations": [
    {
        "name": "External",
        "type": "ecl",
        "request": "launch",
        "protocol": "http",
        "serverAddress": "training.us-hpccsystems-dev.azure.lnrsg.io",
        "port": 8010,
        "path": "",
        "targetCluster": "thor",
        "rejectUnauthorized": true,
        "resultLimit": 100,
        "timeoutSecs": 60,
        "user": "YourNameHere",
        "password": ""
    },
```

{} launch.json ×

ECL IDE



ECL IDE Features:

A full-featured GUI for ECL development providing access to the ECL repository and many of the ECL Watch capabilities.

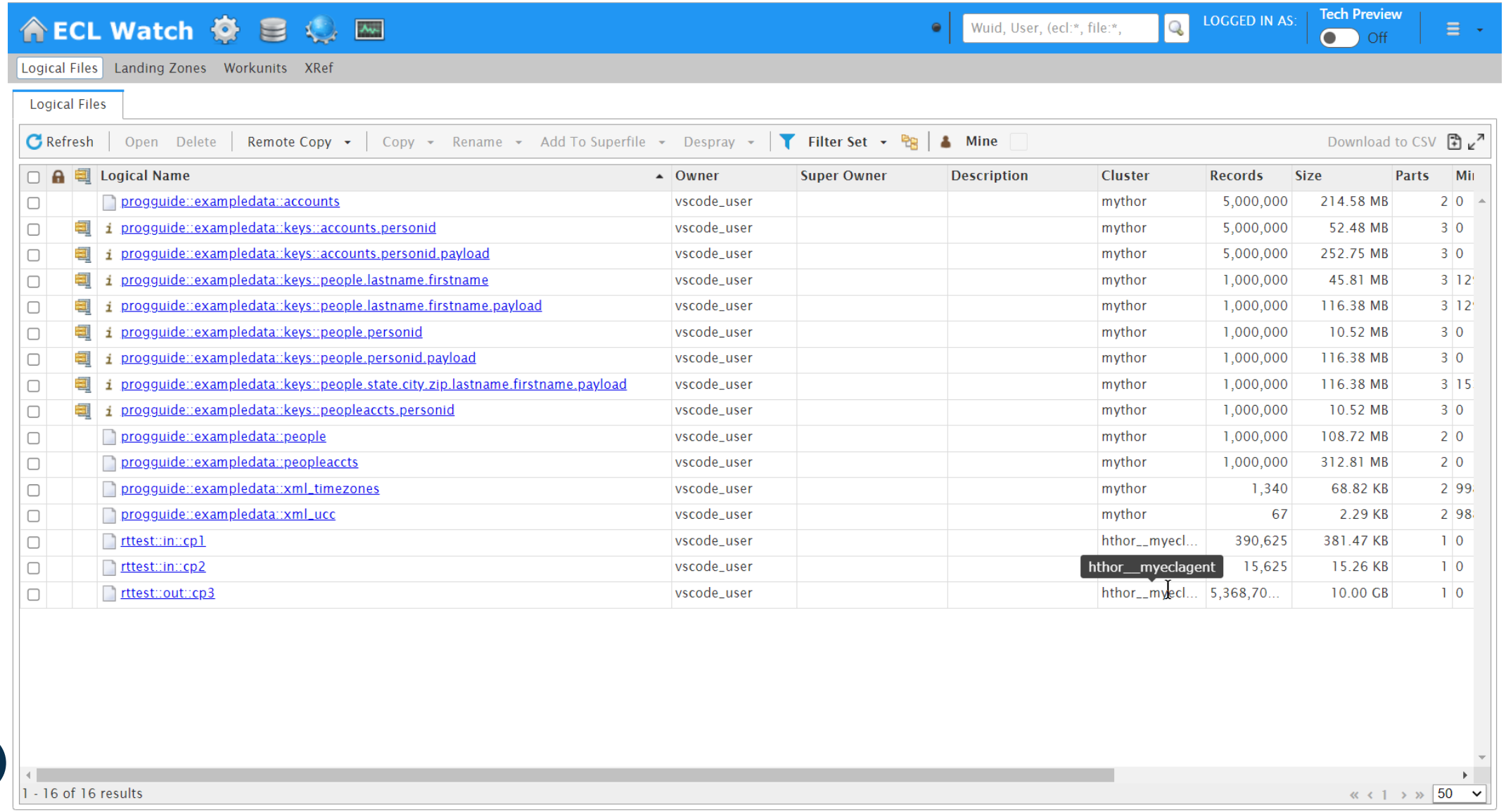
Uses various ESP services via SOAP.

Provides the easiest way to create:

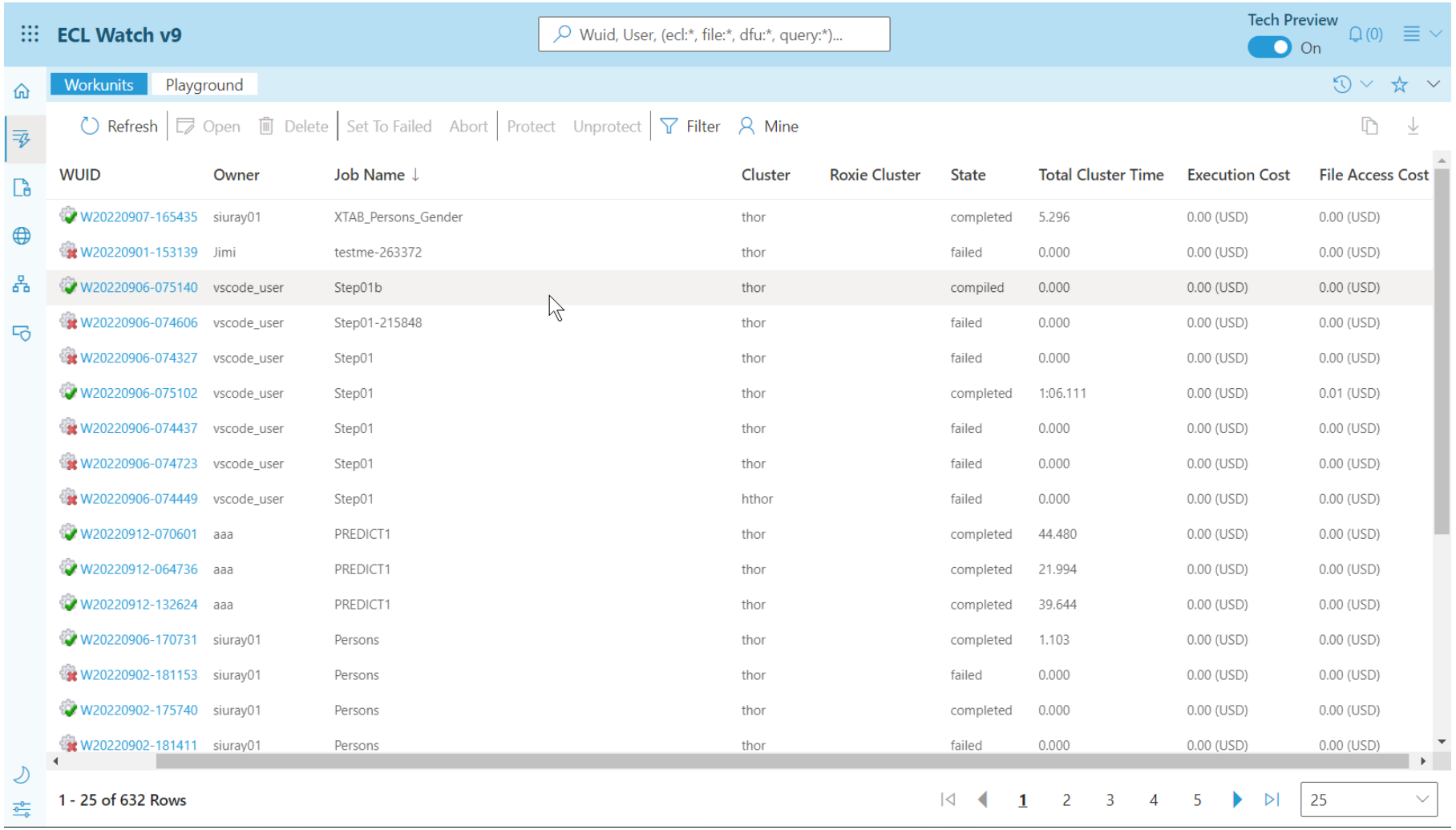
1. Queries into your data.
2. ECL Definitions to build your queries which:

- Are created by coding an expression that defines how some calculation or record set derivation is to be done.
- Once defined, can be used in succeeding ECL definitions.

The ECL Watch (pre-version 9)



The ECL Watch 9



ECL Watch Features:

A web-based query execution, monitoring and file management interface. It can be accessed via ECL IDE or a web browser.

ECL Watch allows you to:

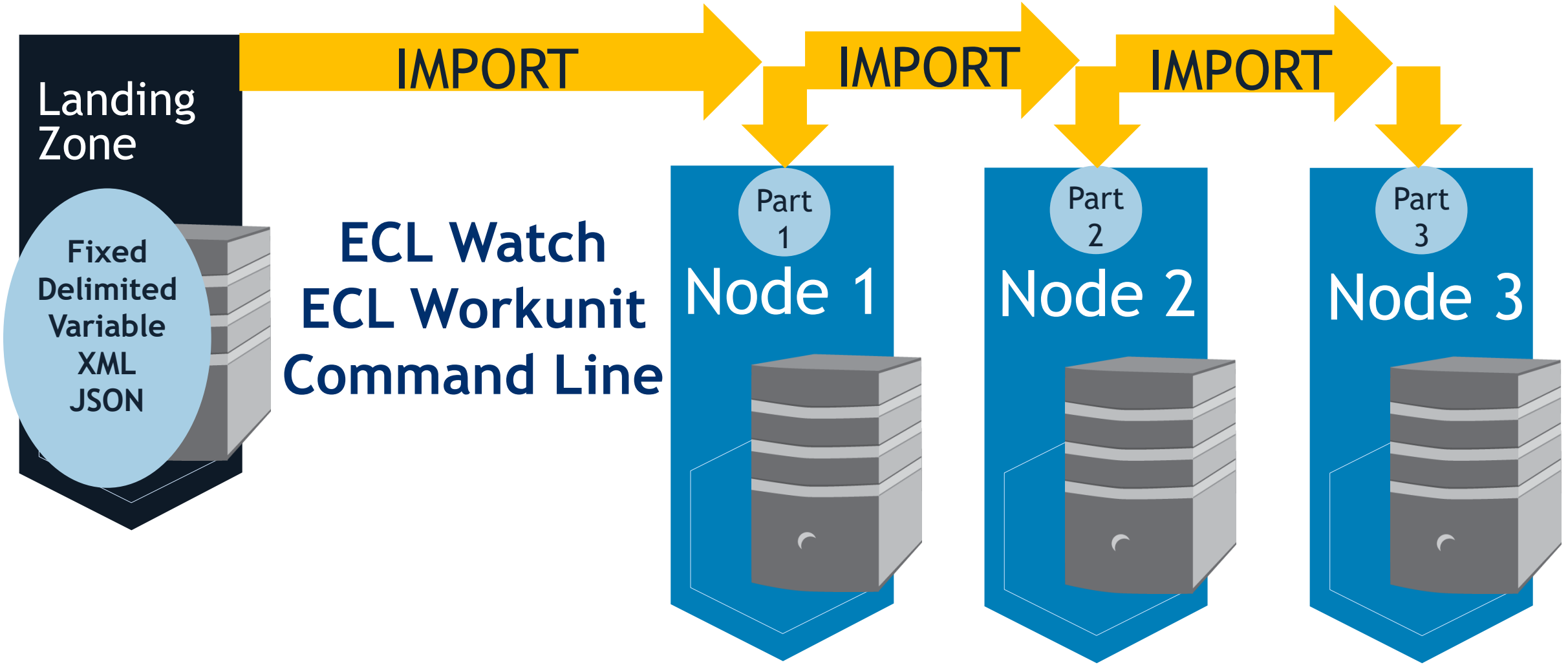
1. See information about active workunits.
2. Monitor cluster activity.
3. Browse through previously submitted Workunits.
4. See a visual representation of the data flow within the WU, complete with statistics which are updated as the job progresses.
5. Search through files and see information including:

- Record counts and layouts.
- Sample records.
- The status of all system servers whether they are in clusters or not.

6. View log files.
7. Start and stop processes.

HPCC SYSTEMS

IMPORT Operation



Treated as a *single* logical file…

ECL Overview

ECL (Enterprise Control Language)

ECL is a language design to query/manipulate massive data and is used for ETL (Extract, Transform, Load) and data visualization.

Extract

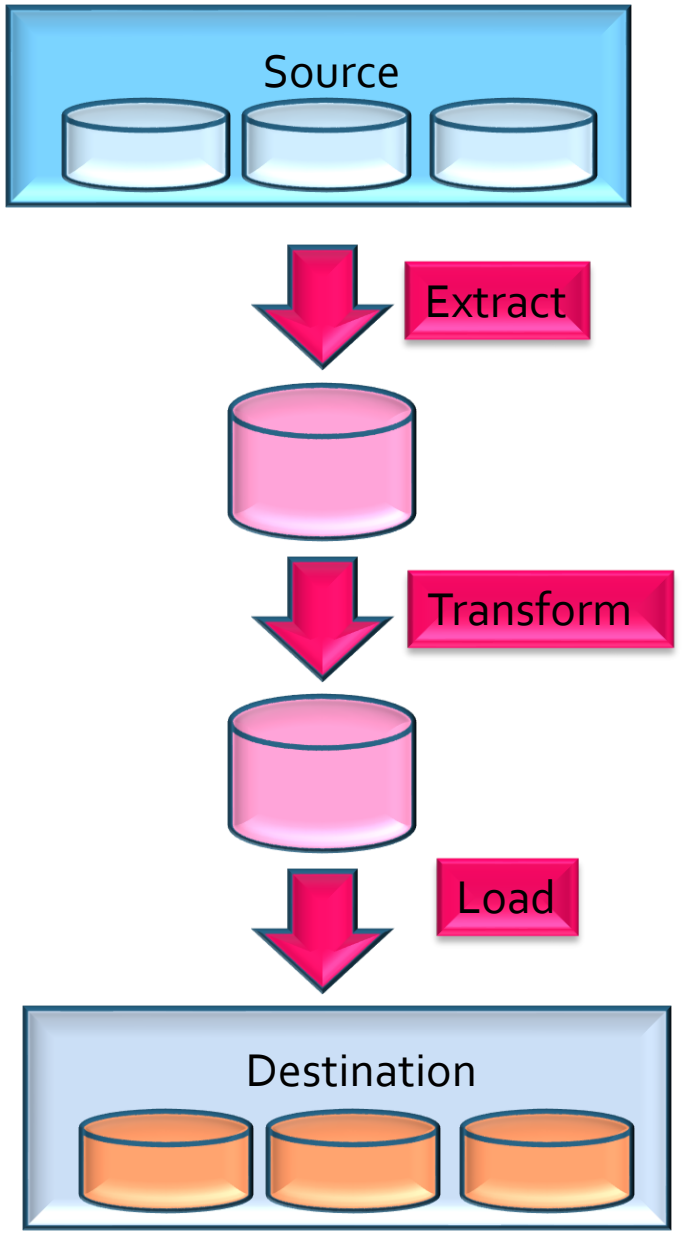
Reading data from different type of datasets

Transform

Formatting/converting data to needed shape

Load

Writing (Delivering) dataset to its target location



Fundamentals of ECL

- ✓ Declarative Language – Made up of Data *Definitions* – Data intensive!
- ✓ **Not** case-sensitive
- ✓ White space is ignored (Makes your code more readable)

```
// This is a single line comment
/* A block comment */
```

- ✓ Object.Property syntax is used to qualify definition scope and disambiguate field references within datasets:
- ✓ ModuleName.Definition //reference a definition from another module/folder
- ✓ Dataset.Field //reference a field in a dataset or record set

Fundamentals of ECL (Continued)

- ✓ Definition assignment is **:=**
- ✓ Semicolon terminator: `num := 12;`
- ✓ Equality test is = `valOne = valTwo`
- ✓ Not equal: Use **<>** or **!=**
- ✓ Definitions can be defined only once.
- ✓ Only those definitions that contribute to a result are compiled and used.
- ✓ There are no loops! TRANSFORM and PROJECT is used instead.

HPCC SYSTEMS

Common Data Types

Character

- STRING[n]
- UTF8
- UNICODE[_locale][n]

Numeric

- INTEGER[n]
- UNSIGNED[n]
- REAL[n]
- DECIMAL<n>[_y]
- UDECIMAL<n>[_y]

Other

- BOOLEAN
- SET OF <type>
- RECORD
- DATASET

Usage:

```
Type      Name     := default value
UNSIGNED1 MyNumber := 0;
```

Name must start with a letter and can contain letters, numbers and the underscore character.

HPCC SYSTEMS

Record Structure

Defines the layout of fields in the dataset, order of the fields should be the same as the dataset.

Dataset

A physical data file. It can be defined in code (inline) or can be read from disk.

Job	Catergory	City	State	Avg_Salary
Manager	IT	Atlanta	GA	87000
Director	Art	Atlanta	GA	100000
CIO	IT	Tampa	FL	112000
Sales	General	Chicago	IL	55000

RECORD Structure Example:

```
EXPORT Layout_Company := RECORD
  UNSIGNED  sic_code;
  STRING1   source;
  STRING120 company_name;
  STRING10  prim_range;
  STRING2   predir;
  STRING28  prim_name;
  STRING4   addr_suffix;
  STRING2   postdir;
  STRING5   unit_desig;
  STRING8   sec_range;
  STRING25  city;
  STRING2   state;
  STRING5   zip;
  STRING4   zip4;
  STRING10  phone;
END;
```

HPCC SYSTEMS

DATASET

name **:= DATASET(** ***file***, ***recorddef***, **THOR [***options***]);**
name **:= DATASET(** ***file***, ***recorddef***, **CSV [(** ***options*** **)]);**
name **:= DATASET(** ***file***, ***recorddef***, **XML(** ***path***,[options]));**
name **:= DATASET(** ***file***, ***recorddef***, **JSON(** ***path***,[***options***]));**

- ✓ *name* – The definition name by which the file is subsequently referenced.
- ✓ *file* – A string constant containing the logical filename.
- ✓ *recorddef* – The RECORD structure of the dataset.
- ✓ *options* – options specific to the dataset type.
- ✓ path - A string constant containing the full XPATH to the tag that delimits the records in the *file*
- ✓ *command* – third-party program that creates the dataset.

DATASET introduces a new data file into the system with the specified *recorddef* layout.

HPCC SYSTEMS

RECORDOF

RECORDOF(*recordset*)

- *recordset* – The set of data records whose RECORD structure to use. This may be a DATASET or any derived recordset.

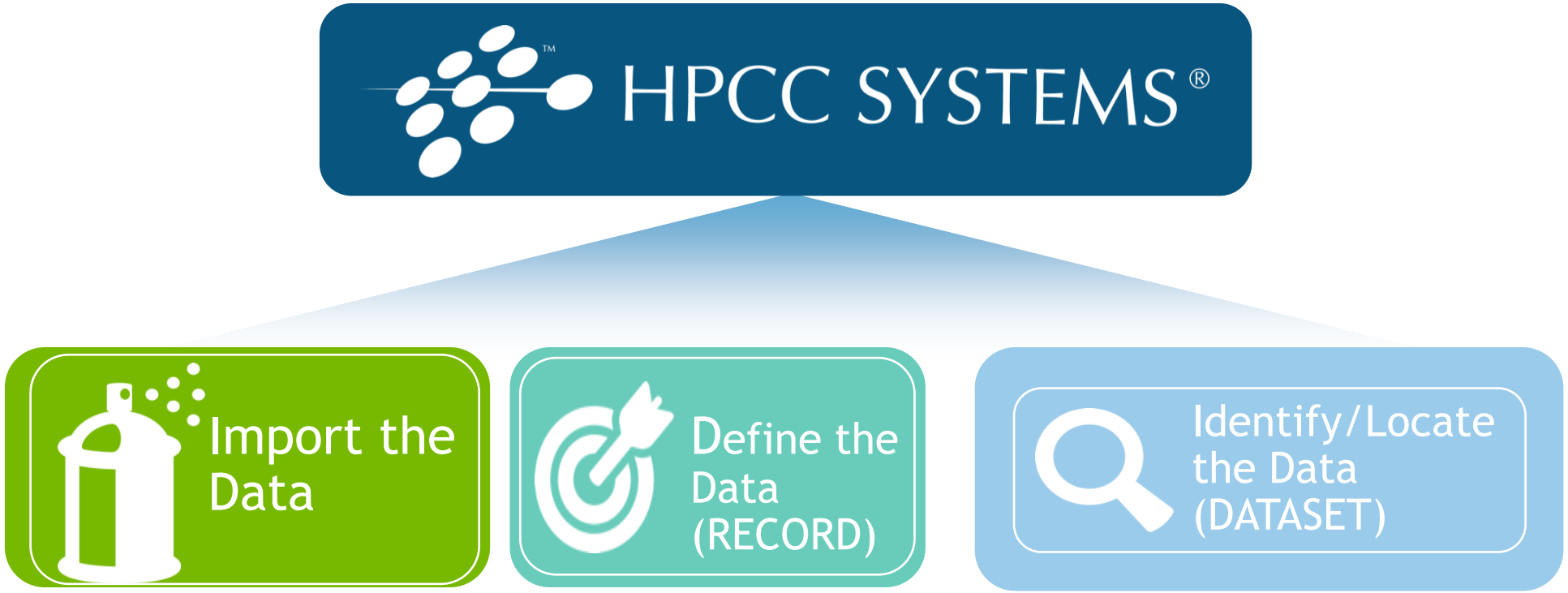
The **RECORDOF** declaration specifies inheriting just the record layout (without default values) of the specified *recordset*.

```
t := TABLE(People,{LastName,FirstName});

r := RECORD
  RECORDOF(t);
  UNSIGNED1 NewByte;
END;
```

HPCC SYSTEMS

Three ECL Data Rules

Before you begin to work on any data in the HPCC cluster, you must always do three things:



RECORD and DATASET example

```
Layout_Company := RECORD
 UNSIGNED     sic_code;
 STRING120    company_name;
 STRING10     prim_range;
 STRING2      predir;
 STRING28     prim_name;
 STRING4      addr_suffix;
 STRING2      postdir;
 STRING5      unit_desig;
 STRING8      sec_range;
 STRING25     city;
 STRING2      state;
 STRING5      zip;
 STRING4      zip4;
END;

EXPORT File_Company_List := DATASET('~CLASS::Company_List', Layout_Company, THOR);
```

Inline DATASET “on the fly” data:

```
SalaryAvg_Layout := RECORD
    STRING Job;
    STRING Category;
    STRING City;
    STRING2 State;
    INTEGER Avg_Salary;
END;

// Inline Dataset
SalaryAvg_DS := DATASET([
                {'Manager', 'IT', 'Atlanta', 'GA', 87000},
                {'Director', 'Art', 'Atlanta', 'GA', 100000},
                {'CIO', 'IT', 'Tampa', 'FL', 112000},
                {'Sales', 'General', 'Chicago', 'IL', 55000}
                ], SalaryAvg_Layout //Layout definition
                );
```

HPCC SYSTEMS

OUTPUT

Let's display the result.

Job	Catergory	City	State	Avg_Salary
Manager	IT	Atlanta	GA	87000
Director	Art	Atlanta	GA	100000
CIO	IT	Tampa	FL	112000
Sales	General	Chicago	IL	55000

CHOOSEN

Returns the first n number of records.

```
// A simple output
OUTPUT(SalaryAvg_DS, NAMED('SalaryAvg_DS'));

//CHOOSEN
OUTPUT(CHOOSEN(SalaryAvg_DS, 2), NAMED('SalaryAvg_Choosen'));
```

##	job	category	city	state	avg_salary
1	Manager	IT	Atlanta	GA	87000
2	Director	Art	Atlanta	GA	100000
3	CIO	IT	Tampa	FL	112000
4	Sales	General	Chicago	IL	55000

##	job	category	city	state	avg_salary
1	Manager	IT	Atlanta	GA	87000
2	Director	Art	Atlanta	GA	100000

HPCC SYSTEMS

SORT

Ascending or descending sort

Job	Catergory	City	State	Avg_Salary
Manager	IT	Atlanta	GA	87000
Director	Art	Atlanta	GA	100000
CIO	IT	Tampa	FL	112000
Sales	General	Chicago	IL	55000

Filter

Choosing a smaller part of dataset. A BOOLEAN expression following any recordset or dataset.

```
//Filter
OUTPUT(SalaryAvg_DS(City = 'Tampa'), NAMED('Tampa_Filter'));

//Sort
SortJobs := SORT(SalaryAvg_DS, Job);
OUTPUT(SortJobs, NAMED('SortJobs'));
```

##	job	category	city	state	avg_salary
1	CIO	IT	Tampa	FL	112000

##	job	category	city	state	avg_salary
1	CIO	IT	Tampa	FL	112000
2	Director	Art	Atlanta	GA	100000
3	Manager	IT	Atlanta	GA	87000
4	Sales	General	Chicago	IL	55000

HPCC SYSTEMS

More on Filtering

All records within *dataset* will be evaluated

If *boolean_expression* evaluates to TRUE for a particular record, it will be included in the result

Logical Operators

- **AND**
- **OR**
- **NOT** or **~**

Comparison Operators

- **=**
- **<>** or **!=**
- **<**
- **>**
- **<=**
- **>=**
- **<=>**

```
youngeOrLowIncome := allPeople(age < 20 OR
                               avgHouseIncome <= 10000);
```

HPCC SYSTEMS

Math Functions

```
MathLayout := RECORD
  INTEGER Num1;
  INTEGER Num2;
  INTEGER Num3;
END;

DS := DATASET([{20,45,34},
               {909,56,45},
               {30,-1,90}],
                  MathLayout);

COUNT(DS);                //Counts  the number records in a dataset -- Returns 3
MAX(DS, Num1);            //Returns the MAX value on a field in a dataset -- Returns  909
MIN(DS, Num2);            //Returns the MIN value on a field in a dataset --  Returns -1
AVE(DS, Num1);            //Returns the AGERAGE  value on a field in a dataset -- Returns 319.6666666666667
SUM(DS, Num1 + Num3);     //Returns the result of adding numbers together -- Returns 1128
TRUNCATE(AVE(DS, Num1)); //Returns the integer portion of the real_value. -- Returns 319
ROUND(3.45);              //Returns the rounded value -- Return 3
ROUND(3.76);              //Returns the rounded value -- Return 4
```

Num1	Num2	Num3
20	45	34
909	56	45
30	-1	90

CORRELATION

NumOne	NumTwo
1	1
2	2
3	3
4	4
5	5
6	6

```
CORRELATION(ds1, NumOne, NumTwo)
```

Returns 1.0

NumObe	NumTwo
1938960000.00	2044820000.00
1779710000.00	854858000.00
2961810000.00	1248480000.00
2774400000.00	1263570000.00
1144160000.00	434290000.00
3387280000.00	1302380000.00
3195380000.00	1711770000.00

```
CORRELATION(ds2, NumOne, NumTwo)
```

Returns 0.497870253554390 8

HPCC SYSTEMS

FUNCTION (ECL Definitions with parameters)

One Line Function

```
INTEGER checkMax (SET OF INTEGER numList) := MAX(numList);

OUTPUT(checkMax([2,5,8,10,45,11]), NAMED('checkMath'));
```

Multi Line Function

```
EXPORT myfunc (STRING val) := FUNCTION
  Result := 'Hello ' + val + ' , welcome to this function';
  RETURN Result;
END;

//Using myfunc
res := myfunc('Jonny');
OUTPUT(res, NAMED('res'));

OUTPUT(myfunc('Sunny'), NAMED('Sunny'));
```

Sunny	Hello Sunny , welcome to this function
res	Hello Jonny , welcome to this function

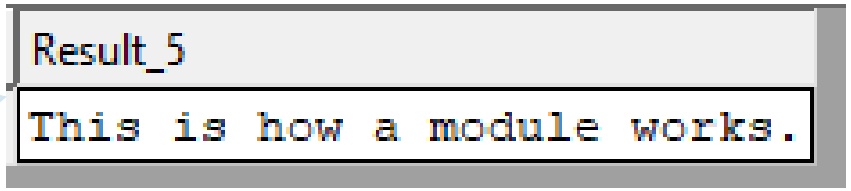
MODULE

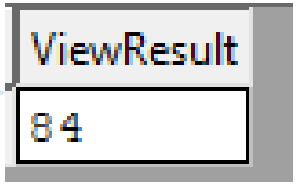
Is a container that allows you to group related definitions.
The *parameters* passed to the module are shared by all the related *members* definitions.

Variable Scope

- Local definitions are visible only up to an EXPORT or SHARED
- SHARED definitions are visible within module.
- EXPORT definitions are visible within and outside of a module .

```
MyMod := MODULE

    // Visible only by MyMod
  SHARED x := 88;
  SHARED y := 42;

    // Visible by MyMod and outsiders
  EXPORT See := 'This is how a module works.';
  EXPORT res := Y * 2;
END;

OUTPUT(MyMod.See);


OUTPUT(MyMod.Res, Named('ViewResult'));
```





HPCC SYSTEMS

TRANSFORM

Specifies exactly how each field in the output record set is to receive its value.

- It should include the result type.
- Should contain name
- Contains parameter list
- SELF: refers to fields in result type.

PROJECT

Processes through all the records in the dataset performing the TRANSFORM.

- LEFT: refers to dataset getting passed to PROJECT.
- COUNTER: Optional counter that counts calls to TRANSFORM

Standalone TRANSFORM

```
Person_Layout := RECORD
    STRING FirstName;
    STRING LastName;
END;

NameDS := DATASET([{'Sun','Shine'},
                   {'Blue','Sun'},
                   {'Silver','Rose'}],
                     Person_Layout);

NameOutRec := RECORD
    STRING FirstName;
    STRING LastName;
    STRING CatValues;
     INTEGER RecCount
END;

NameOutRec CatThem(Person_Layout L, INTEGER C) := TRANSFORM
    SELF.CatValues := L.FirstName + ' ' + L.LastName; //Defines value for new field
    SELF.RecCount := C;  // Adding Counter
    SELF := L;           // Assign everything with same field name from NameDS
END;
CatRecs := PROJECT(NameDS, // Dataset to loop through
                   CatThem  //Transform name
                   (LEFT,   //Left dataset which is NameDS
                   COUNTER  //Simpler Counter
                   ));

OUTPUT(CatRecs, NAMED('CatRecs'));
```

FirstName	LastName
Sun	Shine
Blue	Moon
Silver	Rose

firstname	lastname	catvalues	reccount
Sun	Shine	Sun Shine	1
Blue	Moon	Blue Moon	2
Silver	Rose	Silver Rose	3

NameOutRec: Result Layout

CatThem: Transform Name

Person_Layout: Input Dataset Layout

L : Reference to Person_Layout fields

SELF: Refers to fields in result dataset

C: Will do the Counting

Inline TRANSFORM

```
Person_Layout := RECORD
    INTEGER PersonalID;
    STRING  FirstName;
    STRING  LastName;
END;

NameDS := DATASET([{100, 'Jo', 'Smith'},
                   {203, 'Dan', 'Carpenter'},
                   {498, 'Sally', 'Fryman'},
                   {302, 'Silver', 'Rose'}],
                       Person_Layout);

NameOutRec := RECORD
  INTEGER RecCount;
    INTEGER PersonalID;
    STRING  PersonName;
    STRING  FutureAddress;
END;

CatRecs := PROJECT(NameDS,
                TRANSFORM(NameOutRec,
                    SELF.PersonName := LEFT.FirstName + ' ' + LEFT.LastName;
                    SELF.RecCount   := COUNTER;
                    SELF            := LEFT;
                    SELF            := [];
                     ));
OUTPUT(CatRecs, NAMED('Inline_CatRecs'));
```

PersonalID	FirstName	LastName
100	Jo	Smith
203	Dan	Carpenter
498	Sally	Fryman
302	Silver	Rose

CatRecs: Project Name

NameDS: Input Dataset to loop through

NameOutRec: Result layout

SELF: Refers to fields in result dataset

SELF := LEFT: Assign everything with same field name from NameDS

SELF := []: All unassigned fields will be set to default values

reccount	personalid	personname	futureaddress
1	100	Jo Smith	
2	203	Dan Carpenter	
3	498	Sally Fryman	
4	302	Silver Rose	

TABLE (recordsets in memory, cross-tab tool)

```
Pickup_Layout := RECORD
    STRING10   pickup_date;
    DECIMAL8_2 fare;
    DECIMAL8_2 distance;
END;
Pickup_DS  :=  DATASET([{'2015-01-01', 25.10, 5},
                        {'2015-01-01', 40.15, 8},
                        {'2015-01-02', 30.10, 6},
                        {'2015-01-02', 25.15, 4}],
                            Pickup_Layout);

crossTabLayout  :=  RECORD
   Pickup_DS.pickup_date;
   avgFare   := AVE(GROUP, Pickup_DS.fare);
   totalFare := SUM(GROUP, Pickup_DS.fare);
END;

crossTabDs := TABLE(Pickup_DS, // Input Dataset
                    crossTabLayout,
                    pickup_date);

OUTPUT(crossTabDs, NAMED('crossTabDs'));
```

pickup_date	fare	distance
2015-01-01	25.1	5
2015-01-01	40.15	8
2015-01-02	30.1	6
2015-01-02	25.15	4

pickup_date	avgfare	totalfare
2015-01-01	32.625	65.25
2015-01-02	27.625	55.25

HPCC SYSTEMS

JOIN

The JOIN function produces a result set based on the intersection of two or more datasets or indexes.

INNER: Only those records that exist in both datasets.

LEFT OUTER: At least one record for every record in the left.

RIGHT OUTER: At least one record for every record in the right.

LEFT ONLY: One record for each left record with no match in the left.

RIGHT ONLY: One record for each left record with no match in the right.

FULL ONLY: One record for each left and right record with no match in the opposite.

EmpDS

EmpID	Name	HireYear
1000	Jack	2014
2000	Blue	2016
3000	Mary	2016
5000	Mart	2000
8000	Cat	2002

JobCatDS

EmpID	Department	Title
1000	IT	developer
2000	Biz	Manager
4000	Fin	accountant
8000	IT	analyst

```
InnerJoin := JOIN(EmpDS, JobCatDS,
                  LEFT.EmpID = RIGHT.EmpID,
                  TRANSFORM(EmpResult_Layout,
                      SELF := LEFT,
                      SELF := RIGHT));
```

empid	name	title	department
1000	Jack	developer	IT
2000	Blue	Manager	Biz
8000	Cat	analyst	IT

```
LeftOuterJoin := JOIN(EmpDS, JobCatDS,
                      LEFT.EmpID = RIGHT.EmpID,
                      TRANSFORM(EmpResult_Layout,
                          SELF := LEFT,
                          SELF := RIGHT),
                          LEFT OUTER);
```

empid	name	title	department
3000	Mary		
5000	Mart		

```
FullOuterJoin := JOIN(EmpDS, JobCatDS,
                      LEFT.EmpID = RIGHT.EmpID,
                      TRANSFORM(EmpResult_Layout,
                          SELF := LEFT,
                          SELF := RIGHT),
                          FULL OUTER);
```

empid	name	title	department
1000	Jack	developer	IT
2000	Blue	Manager	Biz
3000	Mary		
0		accountant	Fin
5000	Mart		
8000	Cat	analyst	IT

HPCC SYSTEMS

VISUALIZATION (built-ins and an ECL Bundle)

Methods include

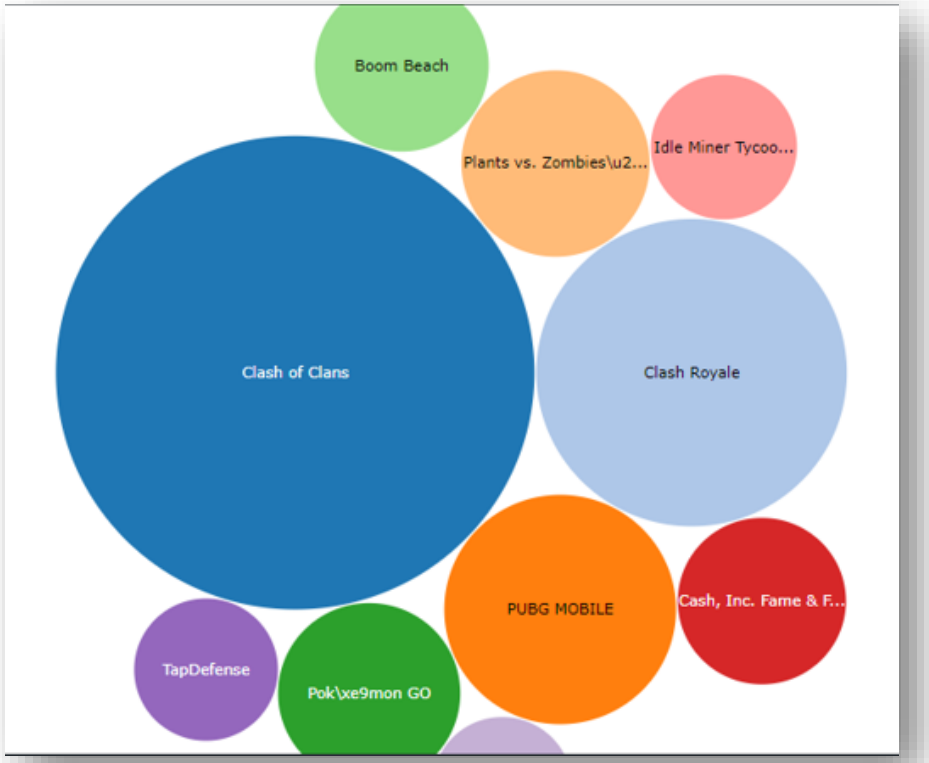
- Two-Dimensional
- Multi-Dimensional Methods
- Geospatial
- General

A basic visualization typically requires the following steps:

1. Creation of a suitable dataset.
2. Output the dataset with a suitable name, so that visualization can locate the data.
3. Create (and output) the visualization, referencing the named output from step 2

```
top_user_rating_count := TOPN(
                    TABLE(clean_mod.games_ds,
                            {name,
                            user_rating_count}) ,
                            10,
                            -user_rating_count);


OUTPUT(analysis_mod.top_user_rating_count, NAMED('user_rating_count'));
Visualizer.TwoD.Bubble('user_rating_count',
                    /*datasource*/,
                     'user_rating_count');
```

- Bubble
- Pie
- Bar
- Scatter
- Line
- WorldCloud
- Area



Useful links!

NSU Code Sharks HPCC Systems Wiki Page:
https://wiki.hpccsystems.com/display/hpcc/Nova+Southeastern+University+-+Code+Sharks+2

Learn ECL Portal:
https://hpccsystems-solutions-lab.github.io

ECL documentation
https://cdn.hpccsystems.com/releases/CE-Candidate-9.4.4/docs/EN_US/ECLLanguageReference_EN_US-9. 4.4-1.pdf

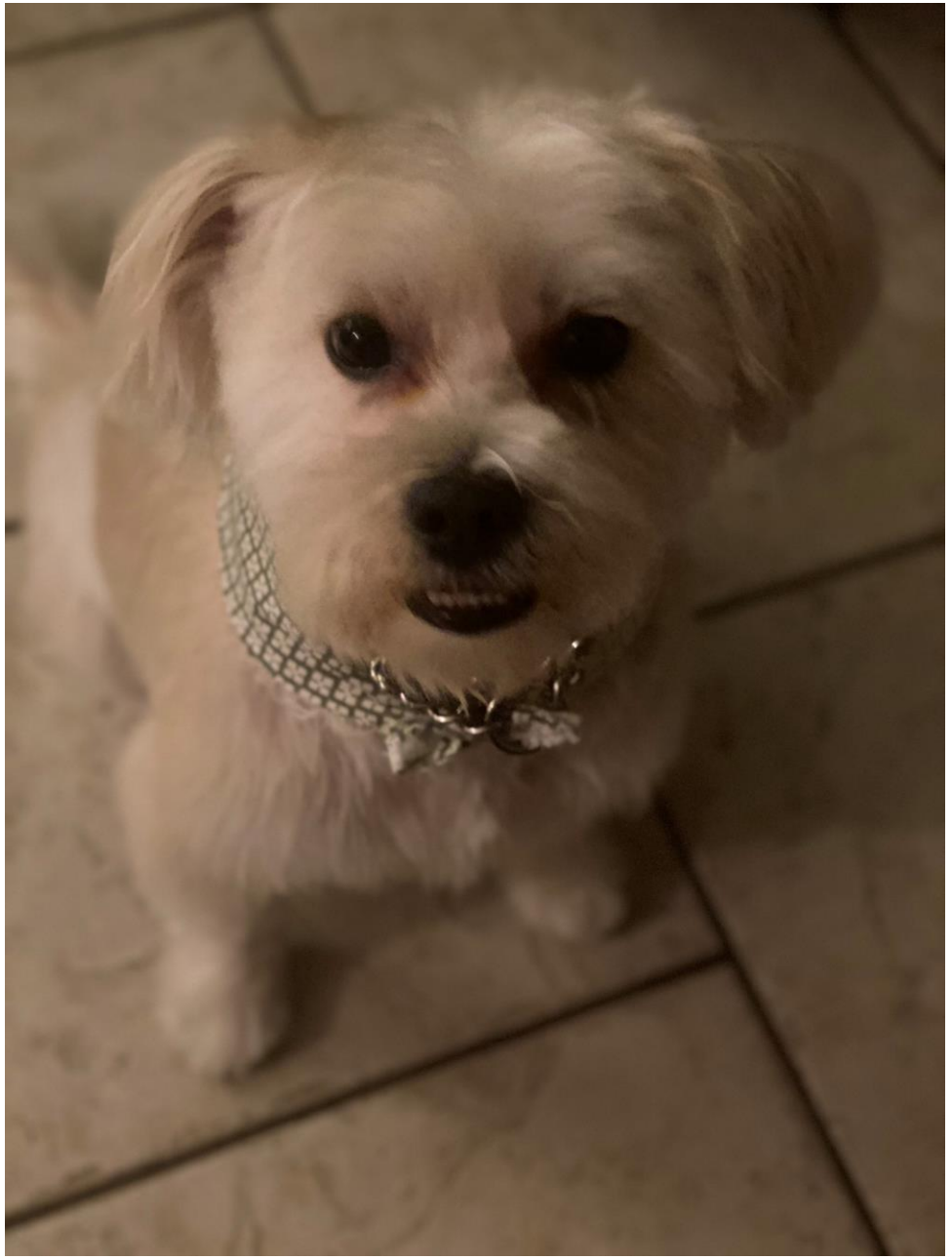
Visualization document
https://cdn.hpccsystems.com/releases/CE-Candidate-9. 4.4/docs/EN_US/VisualizingECL_EN_US-9. 4.4-1.pdf

Standard Library
https://cdn.hpccsystems.com/releases/CE-Candidate-9. 4.4/docs/EN_US/ECLStandardLibraryReference_EN_US-9. 4.4-1.pdf

Machine Learning
https://hpccsystems.com/download/free-modules/machine-learning-library

Get in Touch after the Challenge!

Robert.Foreman@lexisnexisrisk.com



HPCC SYSTEMS


HPCC
SYSTEMS