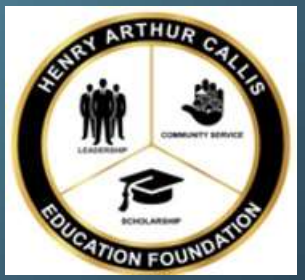


HPCC Systems and Big Data Queries: The Foundations of AI

June 2024

Bob Foreman
Software Engineering Lead
LexisNexis Risk Solutions

Our Company

LexisNexis® Risk Solutions is part of RELX

RELX

RELX is a global provider of information-based analytics and decision tools for professional and business customers, enabling them to make better decisions, get better results and be more productive. RELX serves customers in more than 180 countries and has offices in about 40 countries.

It employs over 36,000 people, over 40% of whom are in North America.

Learn more at www.relx.com

RELX operates in four major market segments:

Scientific, Technical & Medical

ELSEVIER

Risk

LexisNexis® RISK SOLUTIONS

Exhibitions

RX

Legal

LexisNexis® | Legal & Professional

HPCC SYSTEMS™

What we do

We leverage five main capabilities to provide end-to-end solutions that help customers assess risk and opportunity.

Vast Data Resources

We maintain over 12 petabytes of content comprising billions of public and proprietary records.

Big Data Technology

We designed our massively-scalable super-computing platform, HPCC Systems®, enabling us to process at very high speeds – over 270 million transactions per hour.

Advanced Linking

We use our own unique identifier, LexID®, together with a proprietary linking technology. Our patented linking and clustering method is the engine behind many of our products.

Sophisticated Analytics & Insight Capability

We apply data science and leverage patented algorithms, predictive modeling, machine learning and AI to provide data driven solutions and better decision intelligence.

Industry-Specific Expertise & Delivery

The people in our businesses have deep industry experience and expertise – we employ professionals that worked in the industries we serve, so they have walked in the shoes of our customers.

Customer-Focused Solutions

We connect the dots between public records and transactions, resulting in actionable information our customers use to advance their goals.

HPCC SYSTEMS

Our Customers

We work with Fortune 1000 and mid-market clients globally across industries, and federal and state governments.

- **Customers in more than 180 countries**
- **9 out of 10 of the world's top 10 banks**
- **78% of the Fortune 500 companies**
- **98 of the top 100 personal lines insurance carriers**
- **More than 7,500 federal, state and local government agencies**

HPCC SYSTEMS

The Big Data Platform

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

Open Source Technology for Big Data Processing

HPCC Systems®

Born from the deep data analysis experience of LexisNexis Risk Solutions, HPCC Systems is a comprehensive, dedicated cloud-native platform that makes combining data stored in massive, mixed schema data lakes easier and faster. The platform scales very quickly as your data needs grow enabling companies of all sizes to save time and money, now and in the future.

www.hpccsystems.com video

Single, Complete Platform

Two data engines (query and refinery) operating at a high level of speed and accuracy

Cloud-Native Architecture

Automation of Kubernetes makes it easy to set-up, manage and scale your data

Parallel Programming Language (ECL)

High level, data-centric declarative programming language

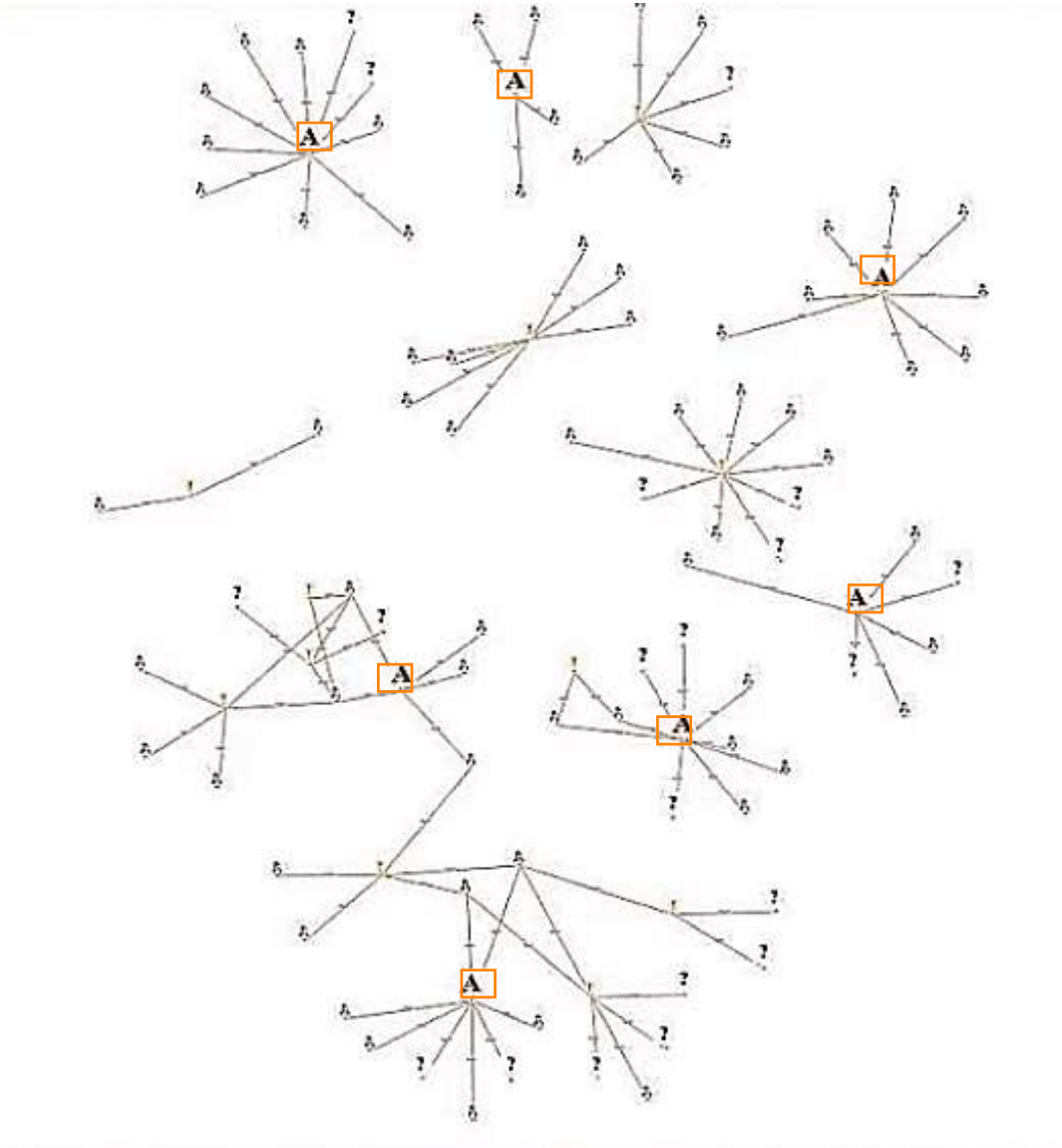
Data Management Tools

Data is easier to manage with robust profiling, cleansing, update and consolidation tools

Machine Learning Library

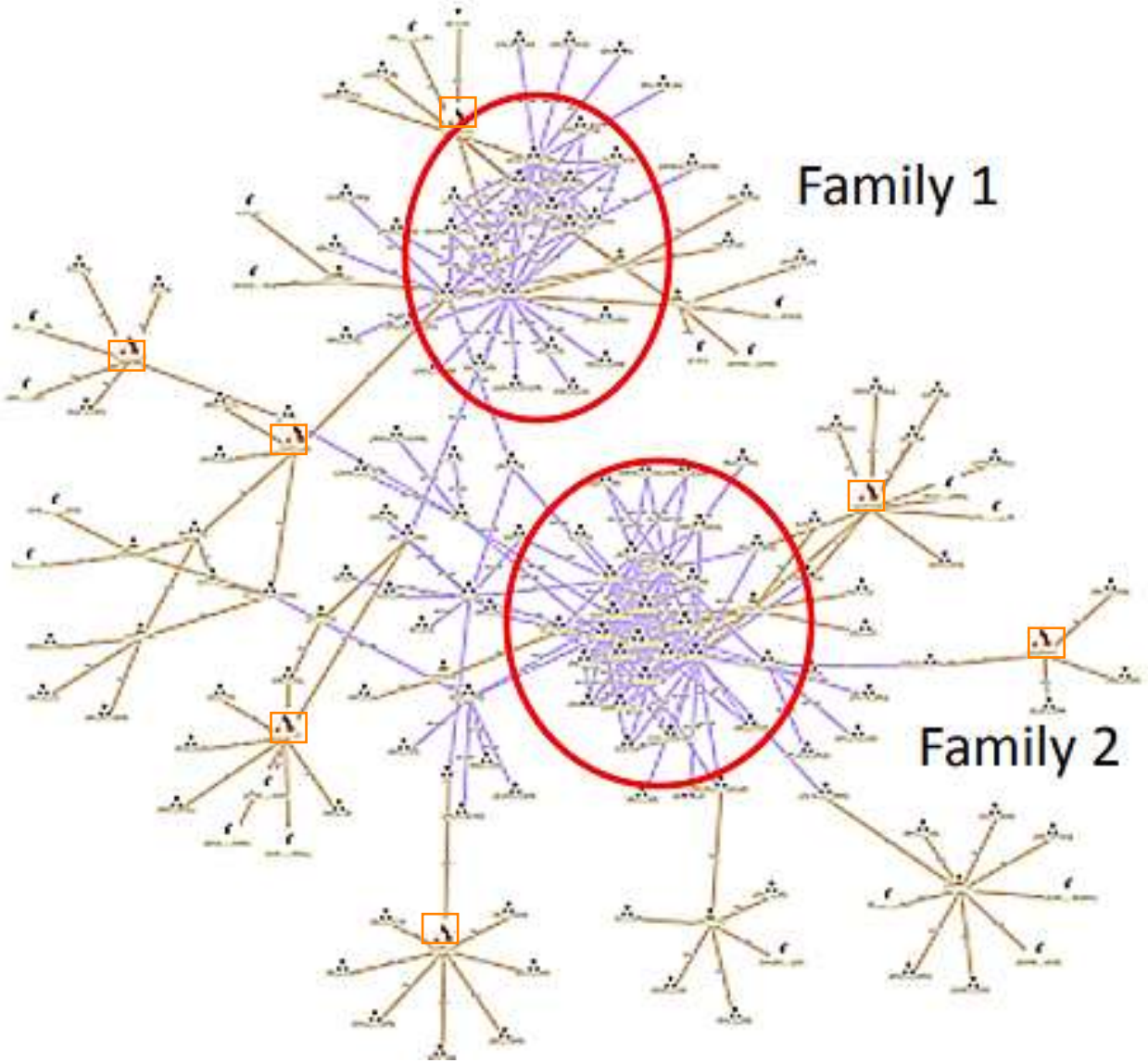
Predictive modeling functionality for machine learning performance

Insurance Fraud Use Case



HPCC SYSTEMS

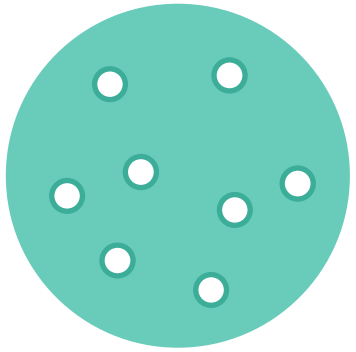
Insurance Fraud Use Case



HPCC SYSTEMS

The Data Centric Approach

A single source of data is insufficient to overcome inaccuracies

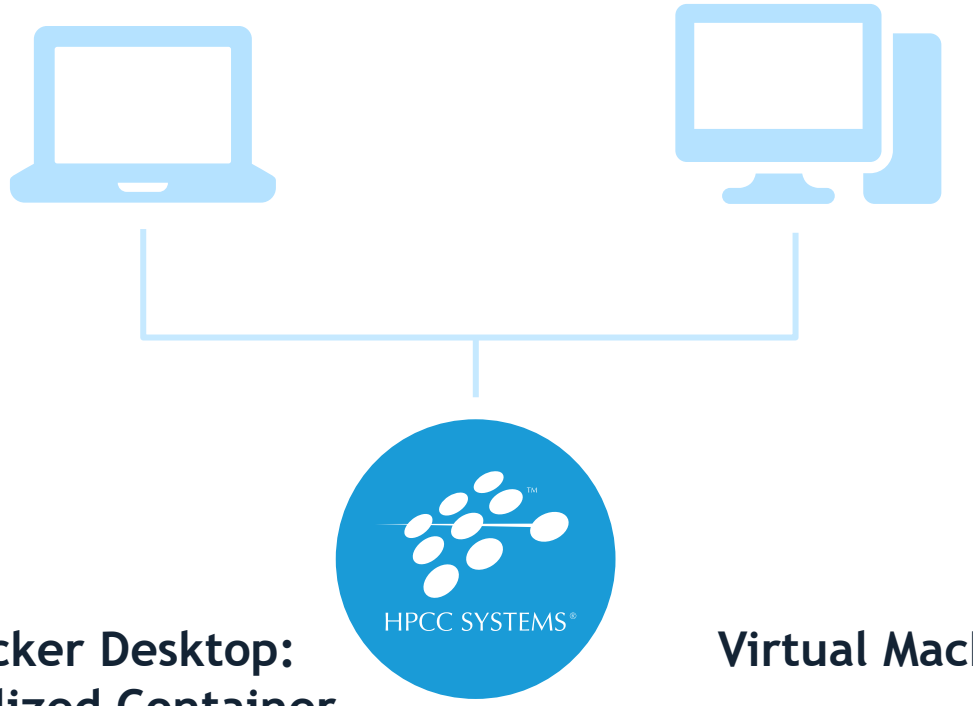


Our platform is built on the premise of absorbing data from **many data sources** and transforming them to **actionable smart data**

HPCC SYSTEMS

Scale from Small to Big

The stack can run on a single laptop or desktop.



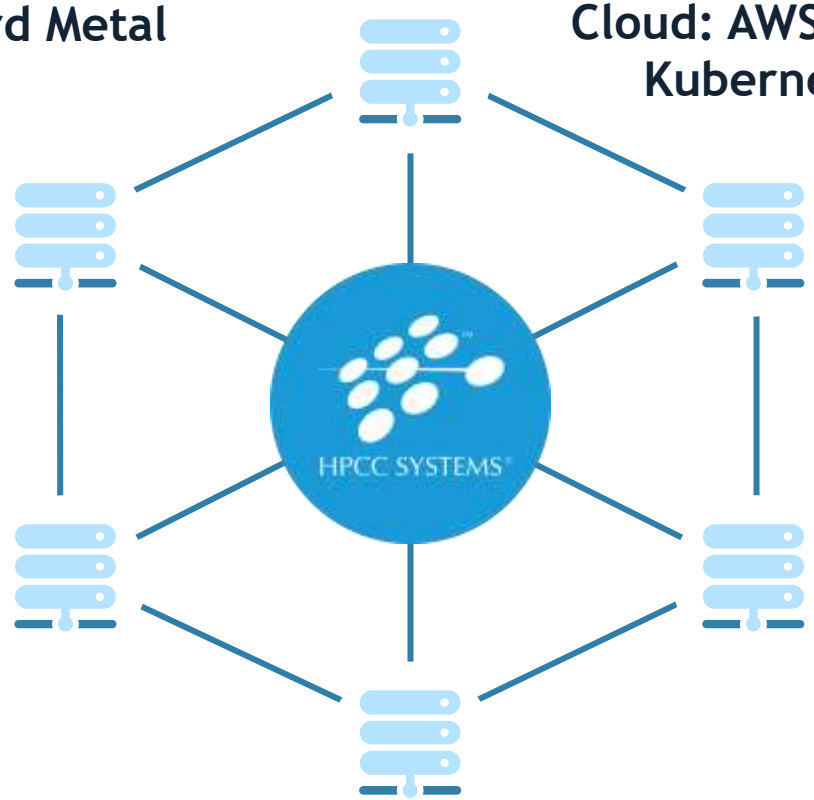
Docker Desktop: Localized Container

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.

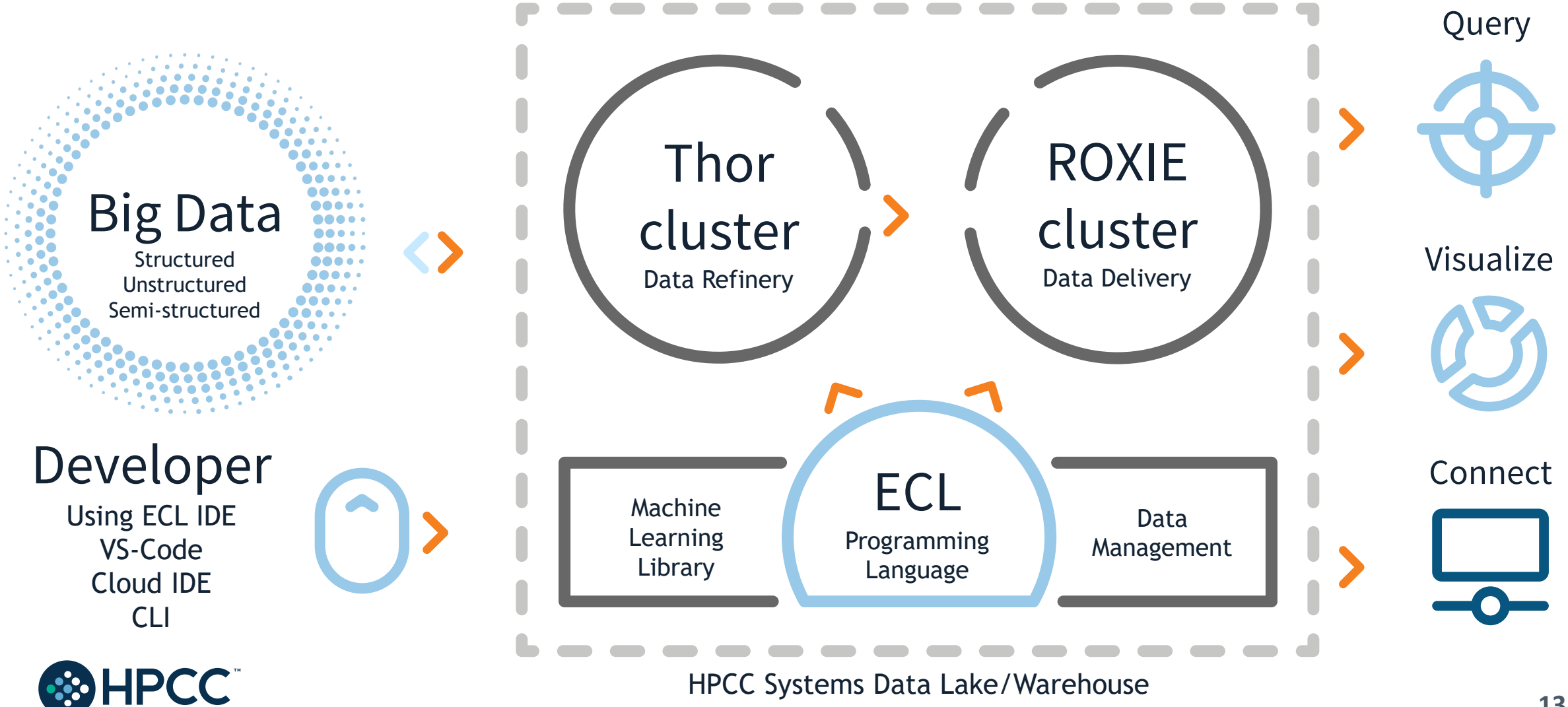
Hard Metal

Cloud: AWS/Azure Kubernetes

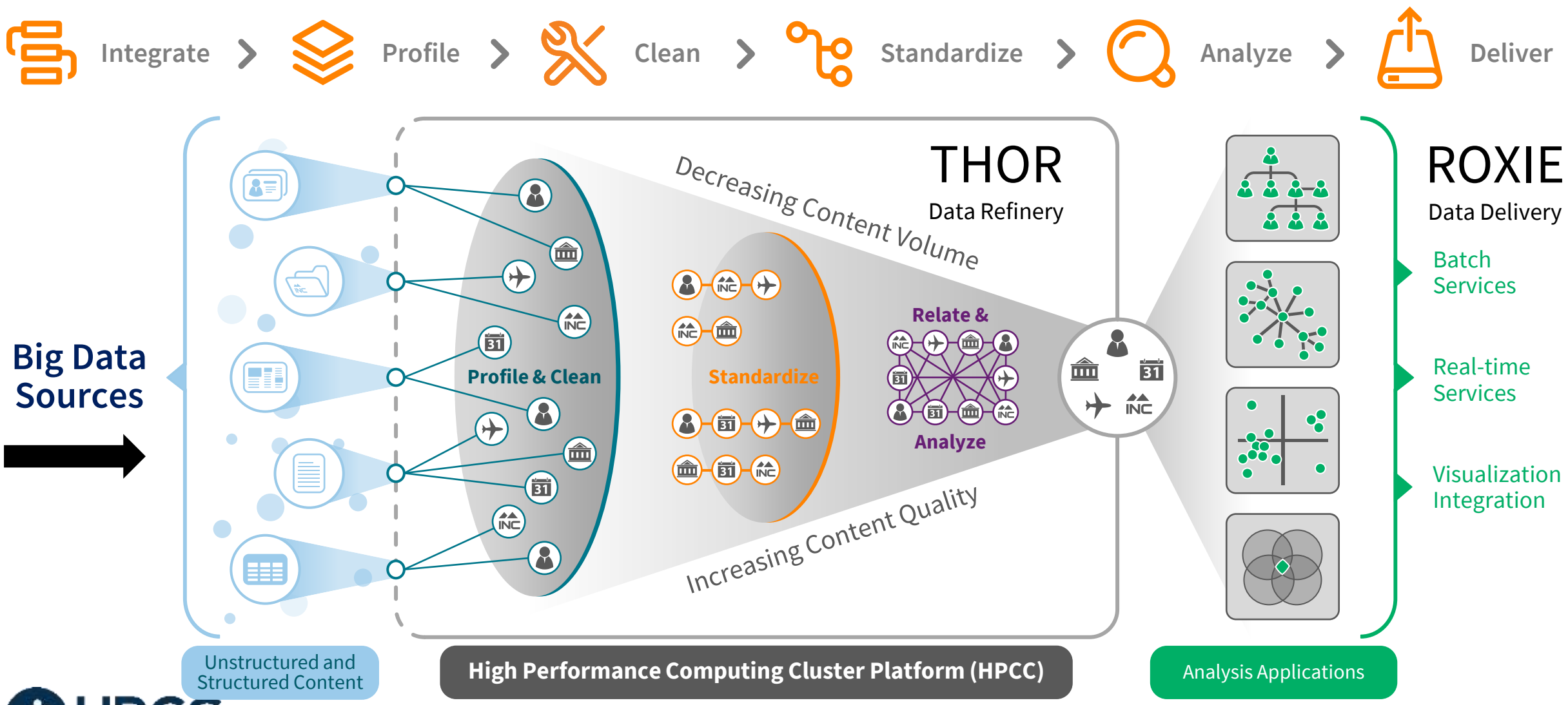


HPCC SYSTEMS

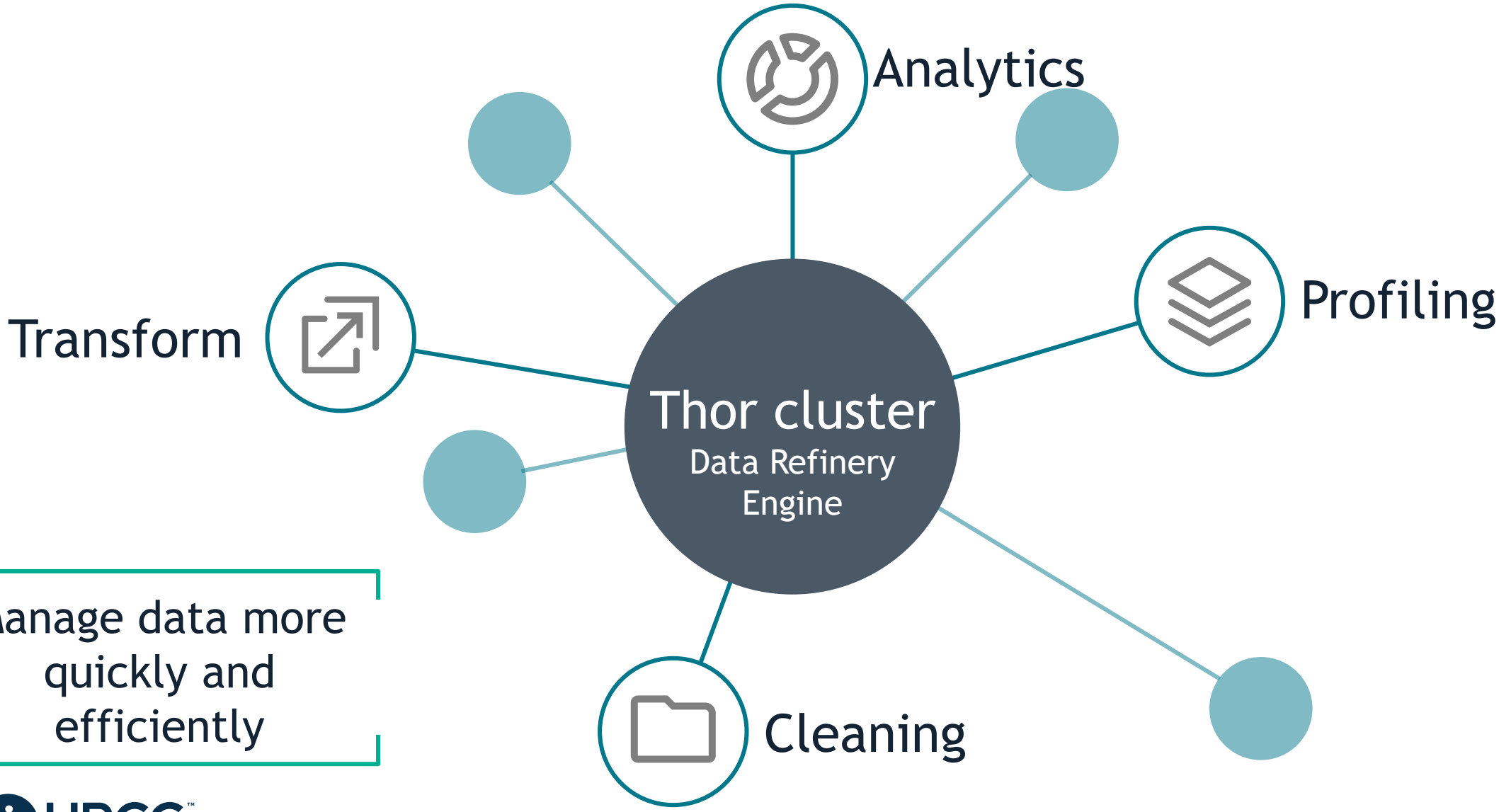
The HPCC Systems Components



HPCC Systems (Small to Big Data) ETL



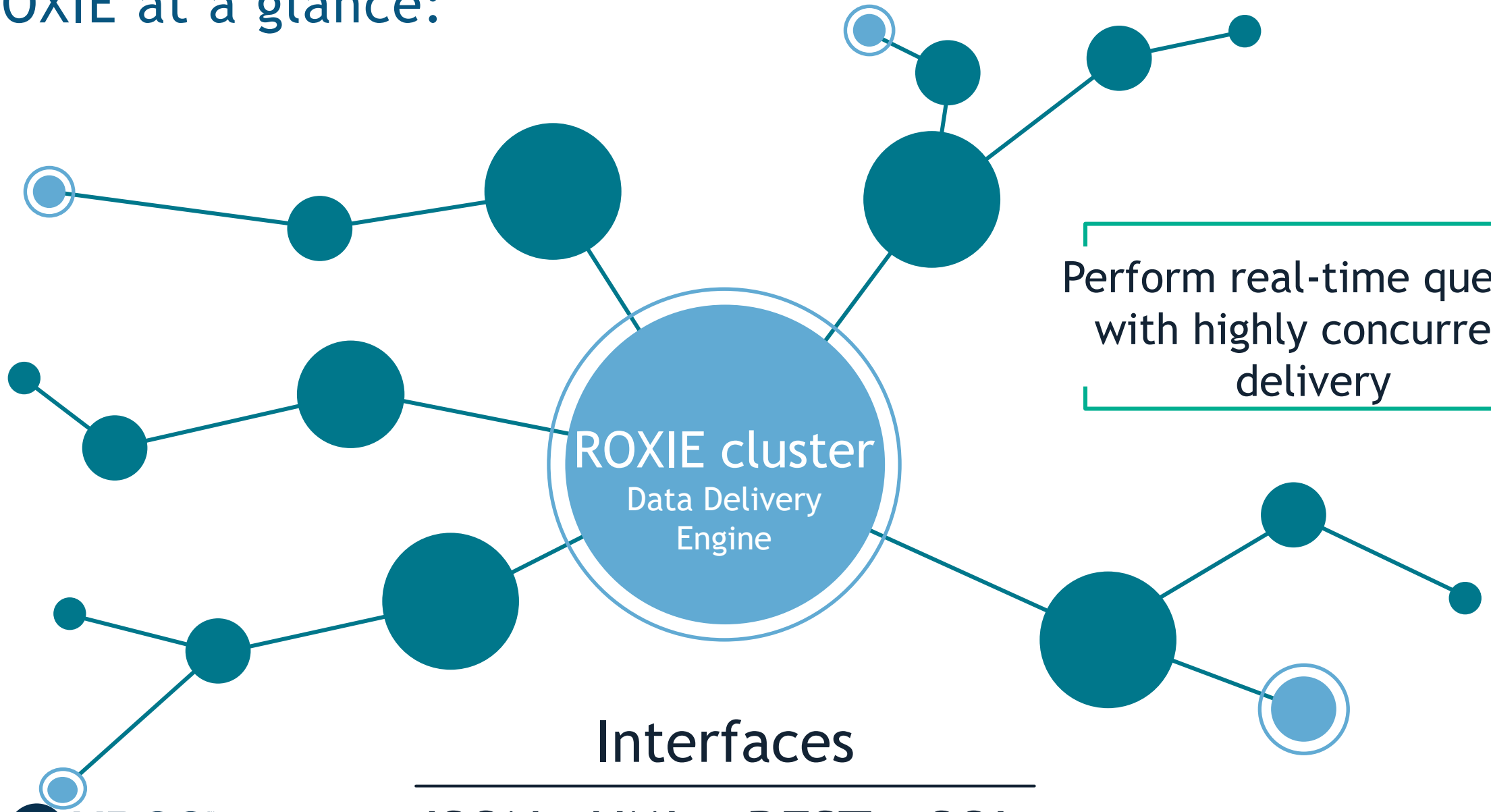
THOR at a glance:



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

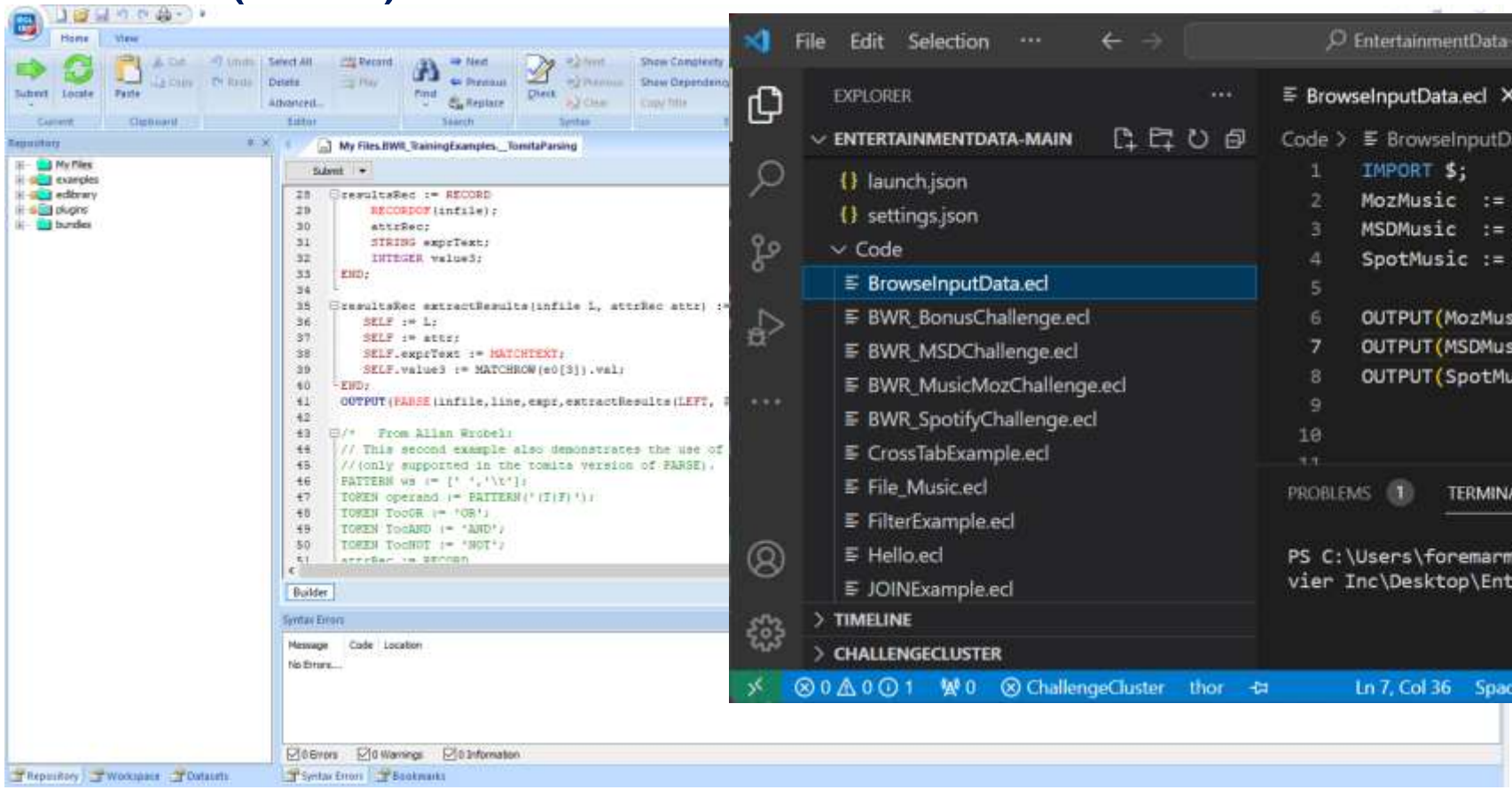
- Optimized for data-intensive operations, declarative, non-procedural and dataflow oriented
- Uses intuitive syntax which is modular, reusable, extensible and highly productive

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:

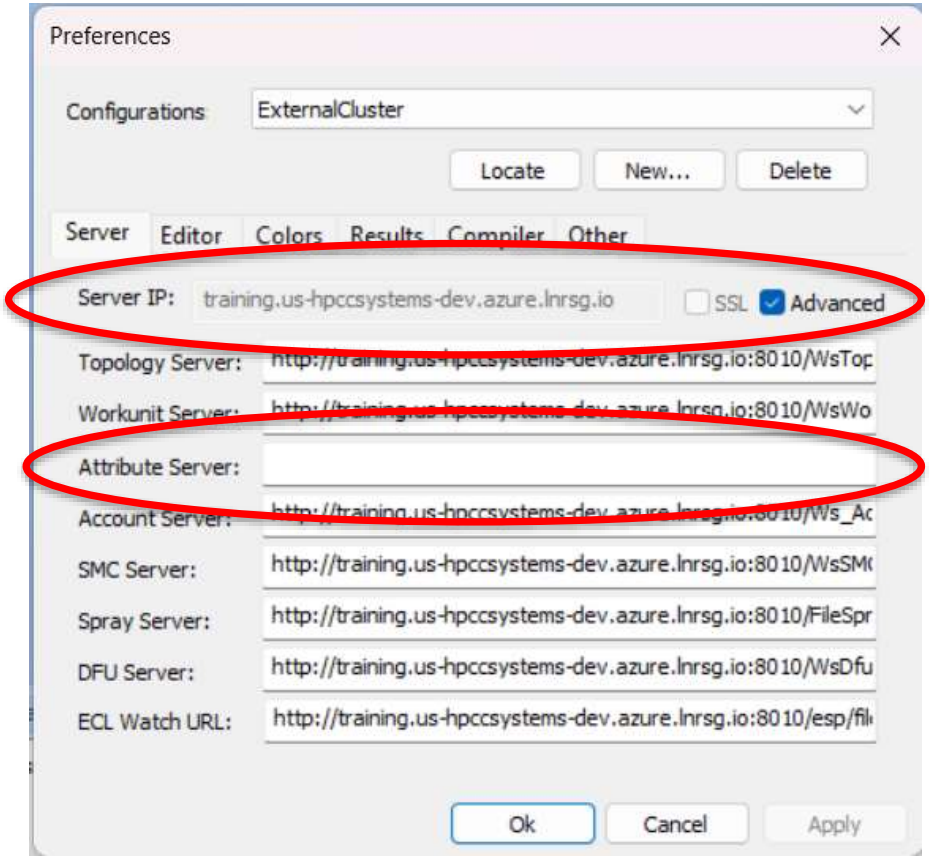
http://training.us-hpccsystems-dev.azure.lnrsg.io:8010/

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.

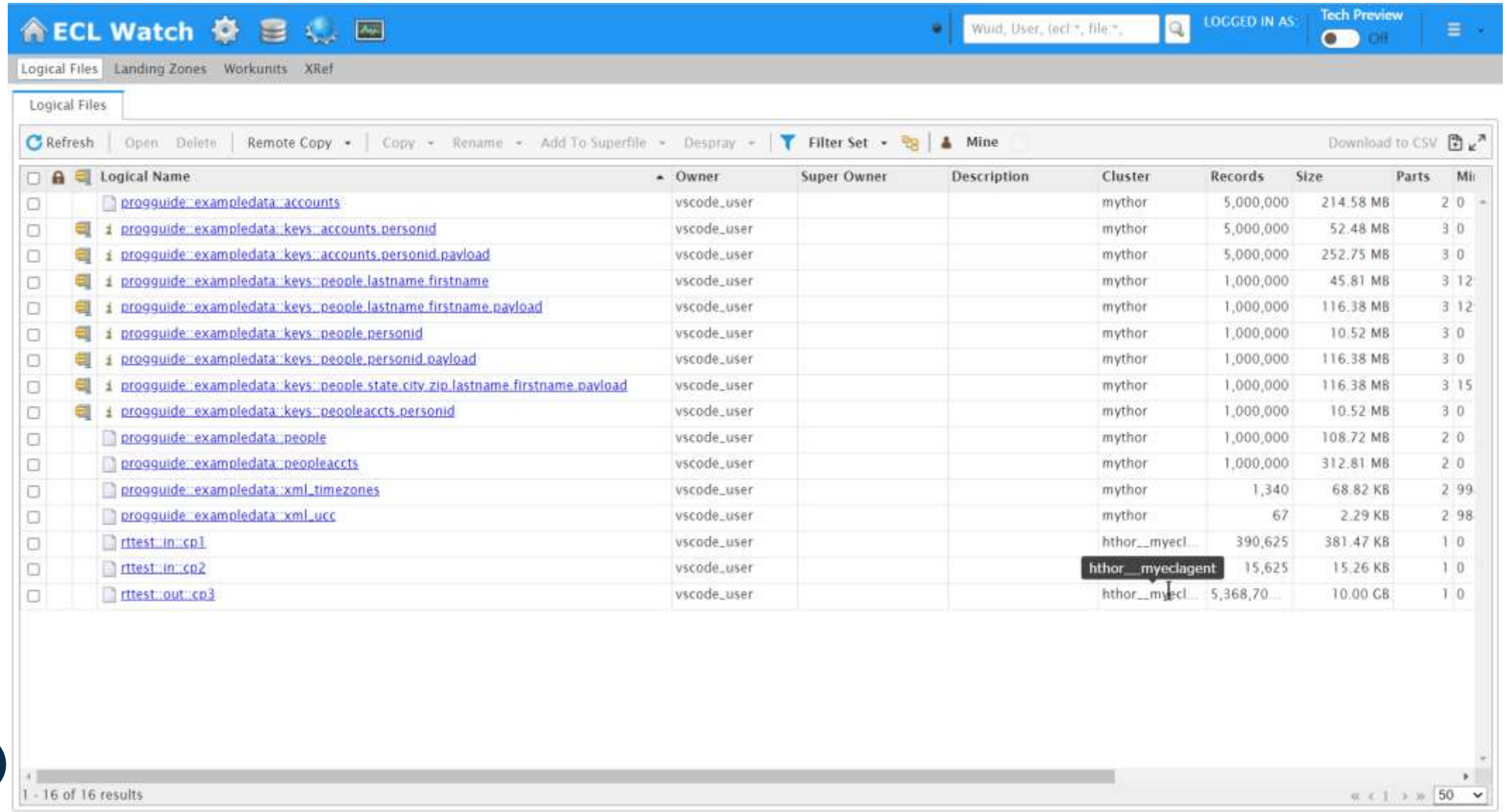
Uses various ESP services via SOAP.

Provides the easiest way to create:

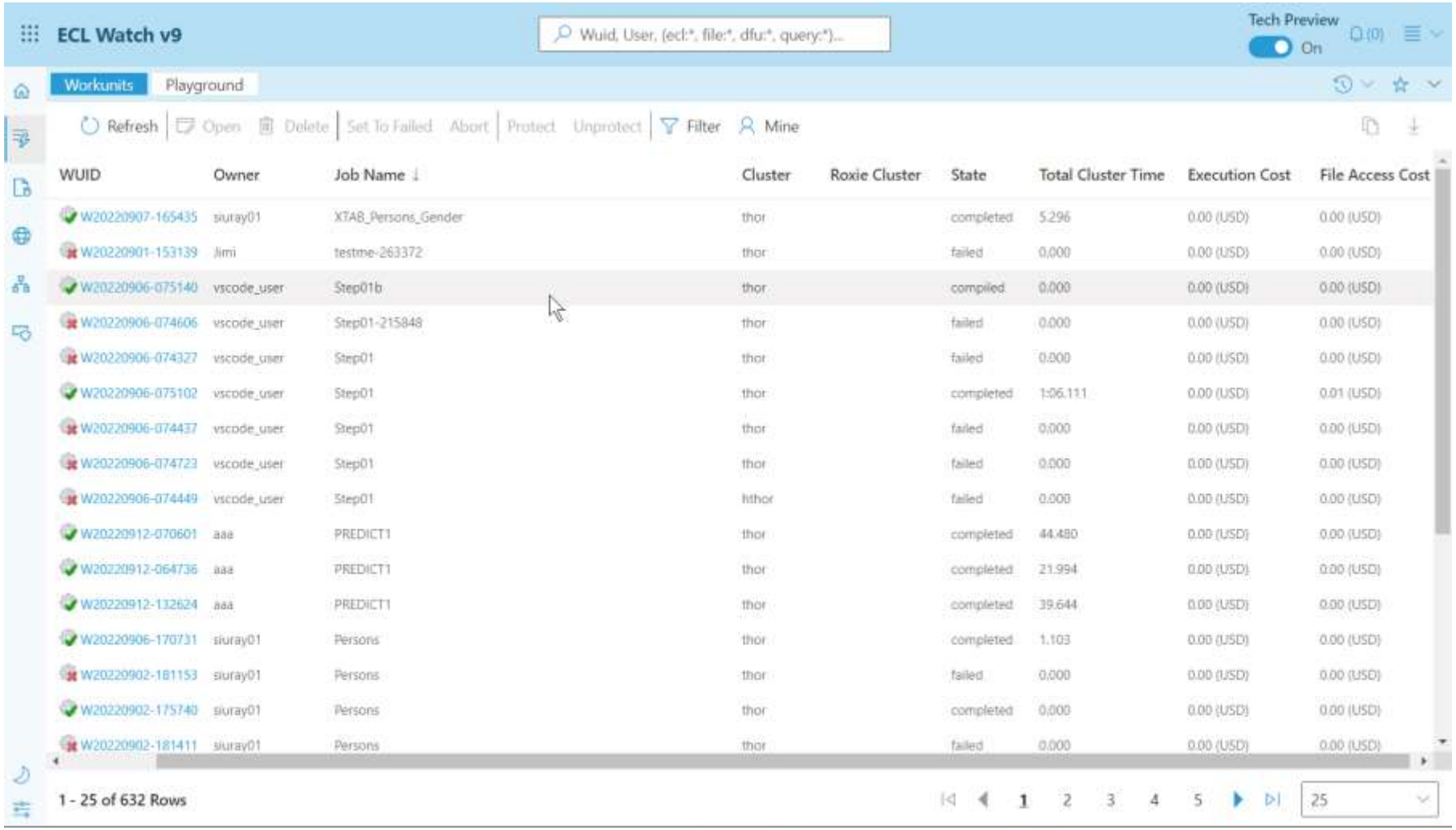
1. Queries into your data.
2. ECL Definitions to build your queries which:

- 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.

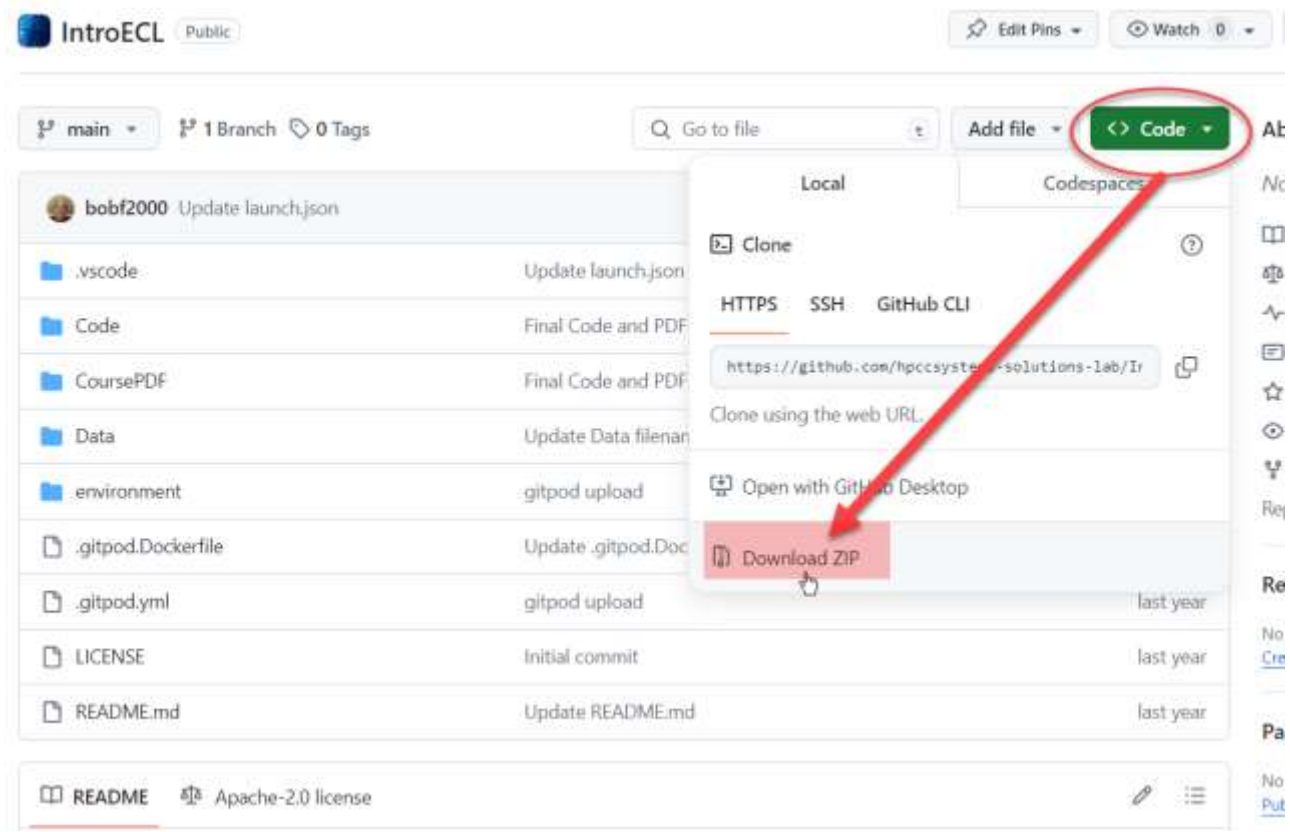
6. View log files.
7. Start and stop processes.

HPCC SYSTEMS

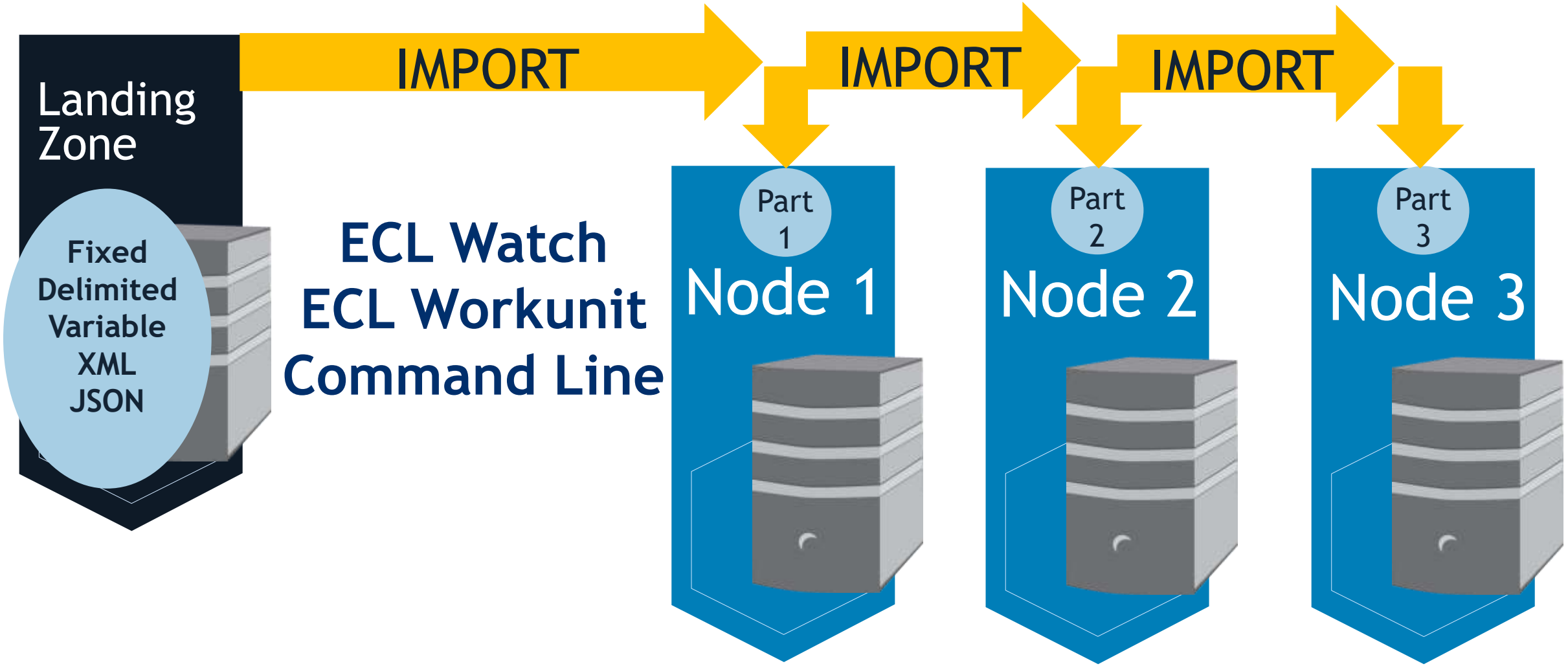
The Repo and an Alternate IDE!

https://github.com/hpccsystems-solutions-lab/IntroECL

Gitpod.io/#https://github.com/hpccsystems-solutions-lab/IntroECL



IMPORT Operation



Treated as a *single* logical file…

The Big Data Language

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

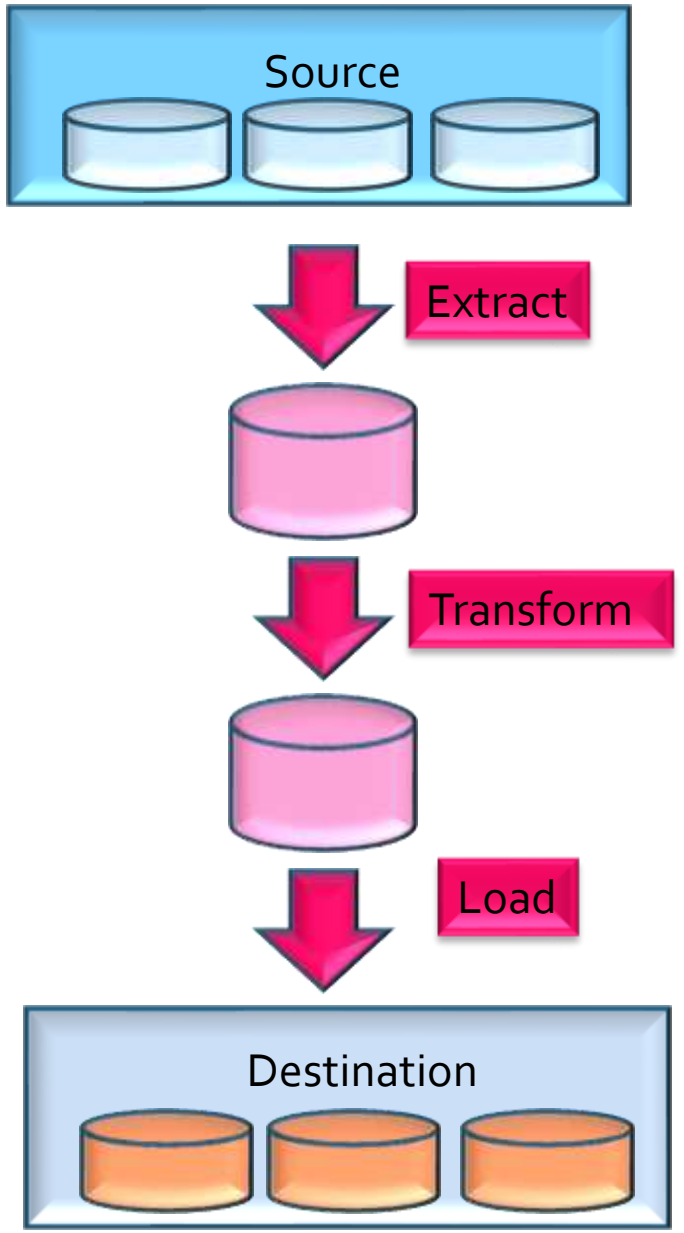
Reading data from different type of datasets

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.

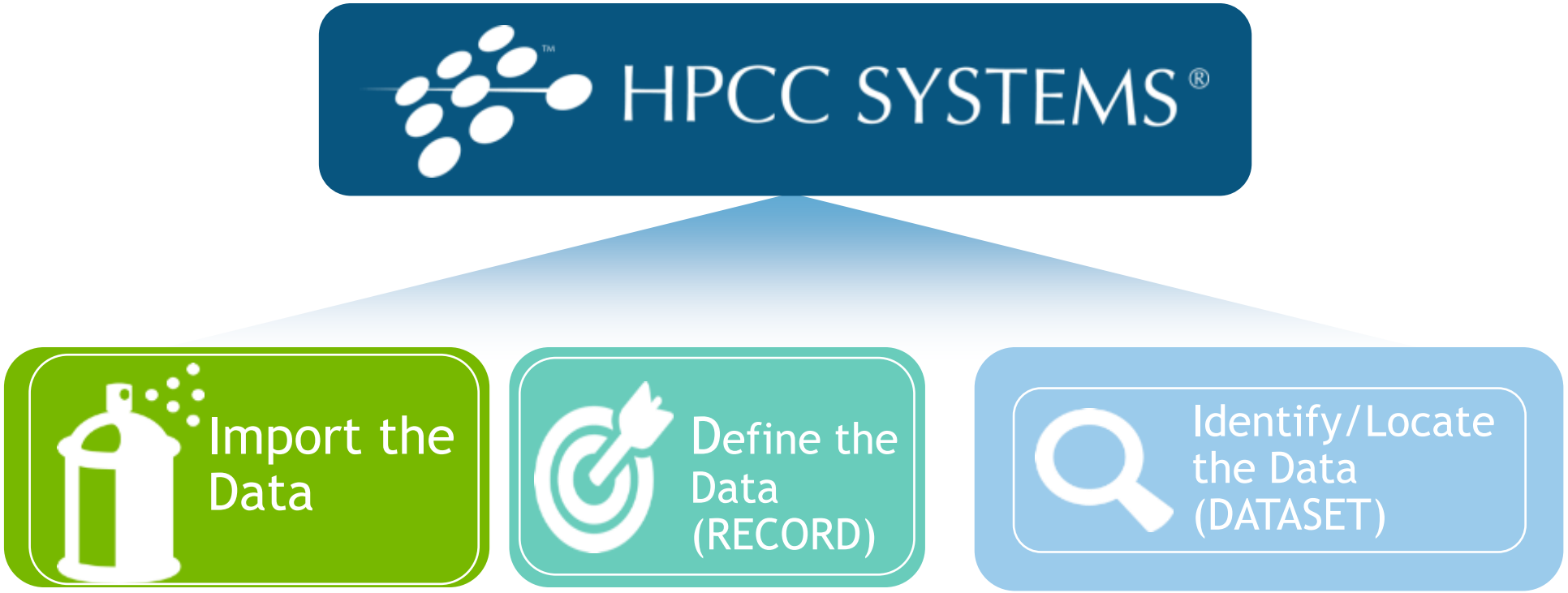
The **RECORDOF** declaration specifies inheriting just the record layout (without default values) of the specified *recordset*.

```
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

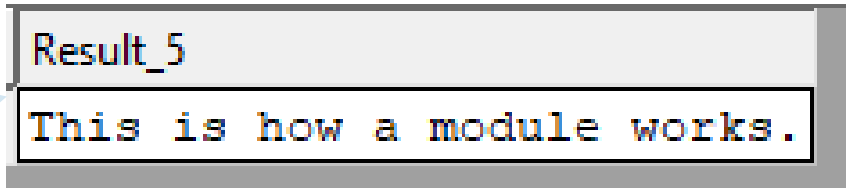
Is a container that allows you to group related definitions.
The *parameters* passed to the module are shared by all the related *members* definitions.

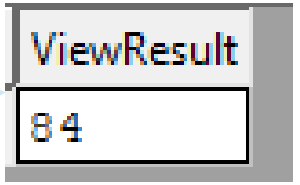
Variable Scope

- 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 .

HPCC SYSTEMS

```
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

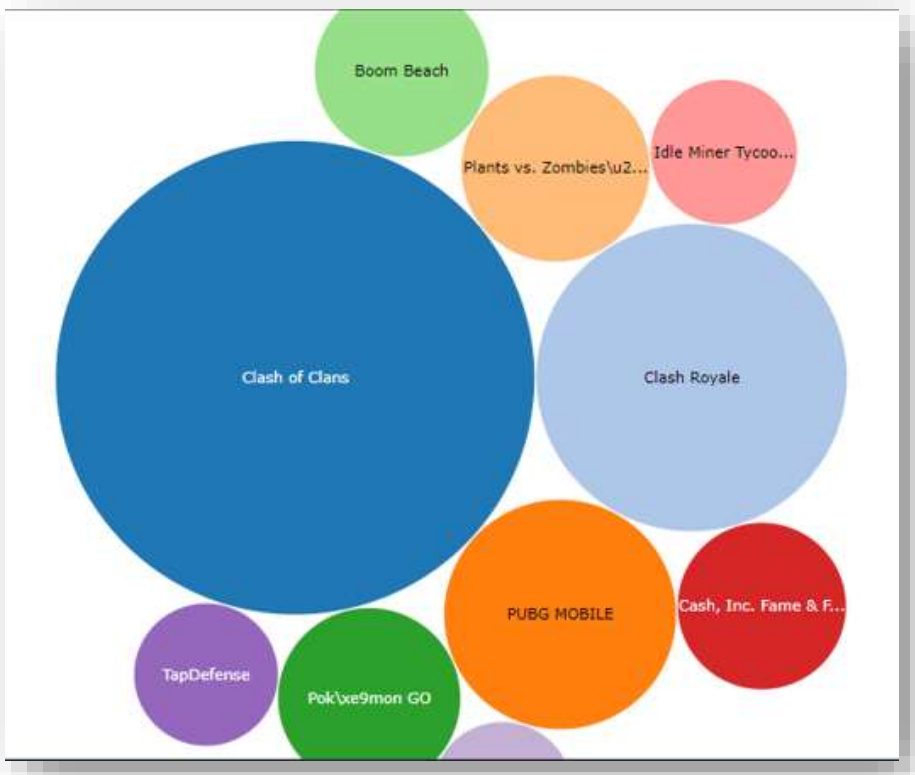
- Two-Dimensional
- Multi-Dimensional Methods
- Geospatial
- General

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



HPCC SYSTEMS

The Million Song Challenge

The Challenge!

Music is Life!

The goal of this challenge is to analyze three music datasets across different genres and discover insights in the data, using the HPCC Systems platform.

You will be presented with several challenge questions in different categories. The more questions you answer, the higher your score will be at the end of the day.

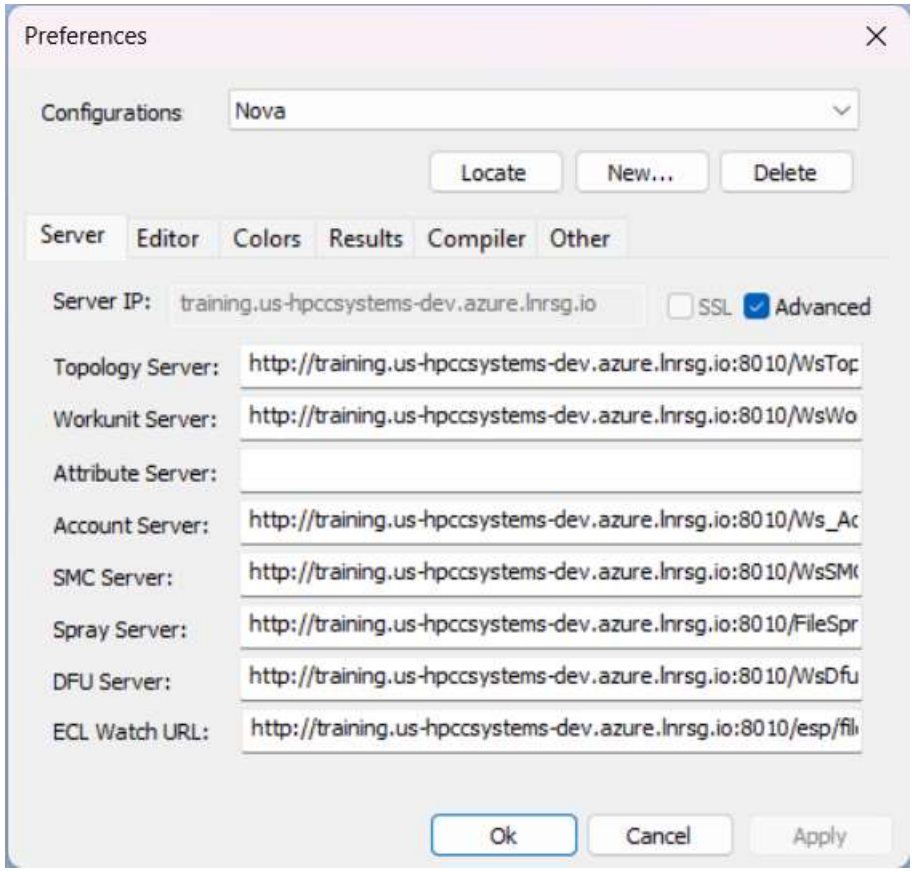
The Playing Field!

HPCC Cluster ECL Watch:

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

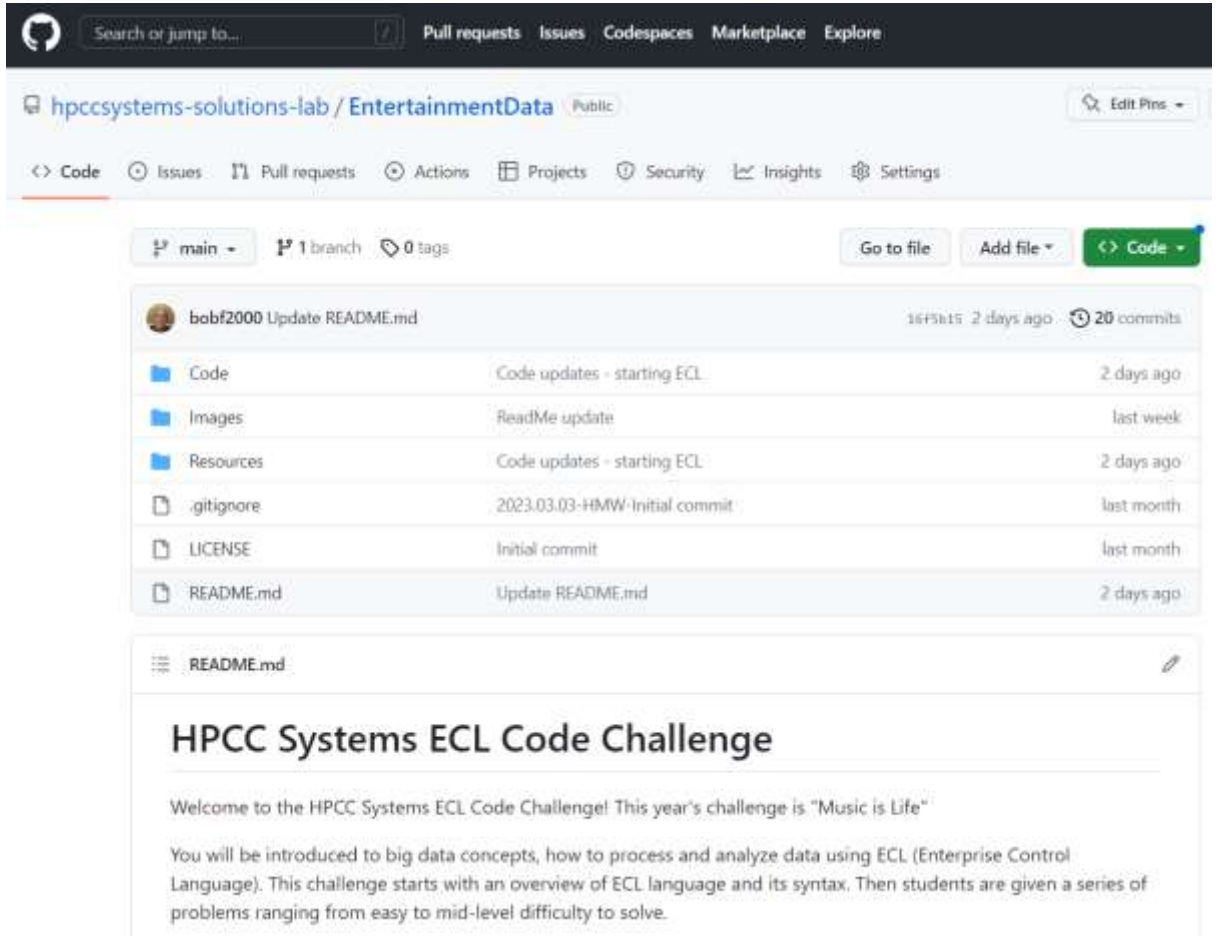
launch.json

```
{
    "name": "Nova",
    "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": ""
},
```



The Repo!

https://github.com/hpccsystems-solutions-lab/EntertainmentData



Million Song Dataset

The Million Song Dataset (MSD) was first created by a company named Echo Nest(which later was acquired by Spotify in 2014). A lot of the data you will see was used as a basis for creating the Spotify search engine.

In this challenge, the original MSD was cleaned and “slimmed down” for this event.

The data dictionary:

RecID	Unique Record ID
song_id	The original song ID used by Echo Nest, not really used in this challenge
title	song title
year	year song was released
song_hotness	download indicator (0 to 1)
artist_id	original artist id from musicbrainz.org
artist_name	artist name
artist_hotness	overall downloads of artist (0 to 1)
familiarity	search indicator of artist
release_id	Album id where song (title) exists

Million Song Dataset (Continued)

release_name	name of release where song exists
latitude	latitude where the song was recorded
Longitude	Longitude where the song was recorded
Location	where the song was recorded
key	Estimation of the key the song in in by Spotify
key_conf	Confidence of the key estimation
loudness	General loudness of the track relative to -60db
mode	Estimation of mode the song is in by Spotify
mode_conf	Confidence of the mode estimation
duration	Song duration in seconds
start_of_fade_out	Fade out of song in seconds
end_of_fade_in	Fade in to song in seconds
tempo	tempo in beats per minute (BPM)
time_signature	number of beats per bar
time_signature_conf	Confidence of the time signature estimation

HPCC SYSTEMS

Million Song Dataset (Continued)

```
CntBars       Total Bars in the song
AvgBarsConf   //Bars_Analysis
BarsConfDev   //Bars_Analysis
AvgBarsStart  //Bars_Analysis
BarsStartDev  //Bars_Analysis
CntBeats      //Beats_Analysis
AvgBeatsConf  //Beats_Analysis
BeatsConfDev  //Beats_Analysis
AvgBeatsStart //Beats_Analysis
BeatsStartDev //Beats_Analysis
```

A bar is one small segment that holds a number of beats.

Multiple beats make up a bar and multiple bars make up a song.

Beats in a bar is dependent on the time signature of the song.

HPCC SYSTEMS

MSD Challenge Questions:

Category One (MS1):

(A) Reverse sort your dataset by "year", and display only the first 50 (HINT: Use SORT, CHOOSEN, and OUTPUT).

(B) Count the total number of records in the dataset (HINT: Use COUNT)

(C) Count the total number of songs released in 2010 and display the first 50 results

(D) How many songs were produced by "Prince" in 1982?

(E) Who sang the song "Into Temptation?"

(F) Sort songs by **Artist** and **Song Title**, and output the first 100

(G)What are the hottest songs by year in the Million Song Dataset? Exclude songs with no year value. HINT: Get the dataset's maximum **song_hotness** value and use it in your output filter.

MSD Challenge Questions:

Category Two (MS2):

(A)Display all songs produced by the artist "Coldplay" that have a "Song Hotness" greater or equal to .75 (>= .75)

•SORT the output by title.

•Also, output the count of the total result

(B)Count all songs whose "Duration" is between 200 AND 250 (inclusive) AND "song_hotness" is not equal to 0 AND "familarity" > .9

(C)Create a new dataset which only has the "Title", "Artist_Name", "Release_Name" and "Year" information.

(D)Calculate Correlation:

•between "song_hotness" AND "artist_hotness“ and between "barsstartdev" AND "beatsstartdev"

HPCC SYSTEMS

MSD Challenge Questions:

Category Three (MS3):

(A) Create a new dataset which only has following conditions

- Column named **Song** that has **Title** values (Song := Title)
- Column named **Artist** that has **artist_name** values (Artist := Artist_Name)
- New BOOLEAN Column called **isPopular**, and it's TRUE is IF **song_hotness** is greater than .80
- New BOOLEAN Column called **IsTooLoud** which is TRUE IF **Loudness** > 0
- Display the first 100
- Result should have 4 columns named **Song**, **Artist**, **isPopular**, and **IsTooLoud**

(B) Display number of songs per **Year** and *count total songs released per year*

- Result has 2 fields, **Year** and **TotalSongs**, verify count is 89

(C) What **Artist** had the overall *hottest* songs between 2006-2007?

- Calculate the average (AVE) **song_hotness** per **Artist_name** for **Year** 2006 and 2007

HPCC SYSTEMS

ECL and Machine Learning

What is Machine Learning?

The broad definition of Machine Learning (from Wikipedia) "the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence.

The "learning" part of Machine Learning (or ML) has several categories, here is what we currently showcase in ECL:

Supervised - The most common type of ML. This method involves the training of the system where the recordsets along with the target output pattern is provided to the system for performing a task.

Unsupervised - This method does not involve the target output which means no training is provided to the system. The system has to learn by its own through determining and adapting according to the structural characteristics in the input patterns.

Deep - Moves into the area of the ML Neural Networks methods. Deep Learning implies multiple layers greater than two (2). Deep also implies techniques use with complex data, like video or audio analysis.

Machine Learning Basics

Let's examine Supervised learning:

Given a set of data samples:

```
Record1: Field1, Field2, Field3, … , FieldM
Record2: Field1, Field2, Field3, … , FieldM
…
RecordN: Field1, Field2, Field3, …, FieldM
```

"Independent" Variables

Note: The fields in the independent data are also known as "features" of the data

And a set of target values,

```
Record1: TargetValue
Record2: TargetValue
…
RecordN: TargetValue
```

"Dependent" Variables

Learn how to predict target values for new samples.

- The set of Independent and Dependent data is known as the "Training Set"
- The encapsulated learning is known as the "Model"
- Each model represents a "Hypothesis" regarding the relationship of the Independent to Dependent variables.
- The hallmark of machine learning is the ability to "Generalize"

HPCC SYSTEMS

Machine Learning Example

Given the following data about trees in a forest:

Height	Diameter	Altitude	Rainfall	Age
50	8	5000	12	80
56	9	4400	10	75
72	12	6500	18	60
47	10	5200	14	53

Learn a Model that approximates Age (the Dependent Variable) from Height, Diameter, Altitude, and Rainfall (the Independent Variables).

It is hard to see from the data how to construct such a model

Machine Learning will automatically construct a model that (usually) minimizes “prediction error”.

In the process, depending on the algorithm, it may also provide insight into the relationships between the independent and dependent variables (inference).

Note: We normally want to map easy to measure (“observed”) features to hard-to-measure (“unobserved”) features.

HPCC SYSTEMS

Quantitative and Qualitative Models

- Supervised Machine Learning supports two major types of model:
  - Quantitative – e.g., Determine the numerical age of a tree
  - Qualitative – e.g., Determine the species of the tree, Determine if the tree is healthy or not.
- Learning a Quantitative model is called "Regression" (a term with archaic origins – don't try to make sense of it).
- Learning a Qualitative model is called "Classification".

Machine Learning Algorithms

- There are many different ML algorithms that all try to do the same things (Classification or Regression)
- Each ML algorithm has limitations on the types of model it can produce. The space of all possible models for an algorithm is called its "Hypothesis Set".
  - "Linear models" assume that the dependent variable is a function of the independent variables multiplied by a coefficient (e.g., f(field1 * coef1 + field2 * coef2 + … + fieldM * coefM + constant))
  - "Tree models" assume that the dependent variable can be determined by asking a hierarchical set of questions about the independent data. (e.g., is height >= 50 feet? If so, is rainfall >= 11 inches?...then age is 65).
  - Some other models (e.g., Neural Nets) are difficult to explain or visualize and are therefore not considered "Explanatory".

Using Machine Learning

- Machine Learning is fun and easy
  - It is simple to invoke one of our ML algorithms, learn a model and make some predictions.
  - If you have some data, give it a try.
  - It will likely provide good insights and may identify some significant possibilities for adding value.

- Machine Learning is also dangerous (not *just* because of robot death rays)
  - There are many ways to produce inaccurate and misleading predictions.
    - Bad questions: yesterday's temperature -> today's stock close
    - Bad data – Missing values, incorrect values -> bad predictions
    - Wrong assumptions regarding the algorithm chosen
    - Insufficient Training Data
    - Overfitting
    - Many others …
  - Always consult a qualified Data Scientist before applying your models to a production activity.

HPCC Platform Machine Learning Approaches

- Embedded Language support for Python and R allow industry ML platforms:
  - Various R packages
  - Sckit-learn (Python)
  - Google TensorFlow (Python)
- Production Bundles
  - Fully supported, Validation tested, HPCC optimized, and Performance profiled.
  - Bundles are generally independent of platform release, and easy to install
  - The Bundle installer will alert if there is a platform version dependency for a particular bundle

ML Core and Supervised Bundles

- ML_Core – Machine Learning Core (https://github.com/hpcc-systems/ML_Core)
  - Provides the base data types and common data analysis methods.
  - Also includes methods to convert between your record-oriented data and the matrix form used by the ML algorithms.
  - This is a dependency for all of the ML bundles.
- PBblas – Parallel Block Basic Linear Algebra Subsystem (https://github.com/hpcc-systems/PBblas)
  - Provides parallelized large-scale matrix operations tailored to the HPCC Platform.
  - This is a dependency for many of the ML bundles.
- LinearRegression – Ordinary least squares linear regression (https://github.com/hpcc-systems/LinearRegression)
  - Allows multiple dependent variables (Multi-variate)
  - Provides a range of analytic methods for interpreting the results
  - Can be useful for testing relationship hypotheses (
- LogisticRegression – Logistic Regression Bundle (https://github.com/hpcc-systems/LogisticRegression)
  - Despite “regression” in its name, provides a linear model-based classification mechanism
  - Binomial (yes/no) classification
  - Multinomial (n-way) classification using Softmax.
  - Allows multiple dependent variables (Multi-variate)
  - Includes analytic methods.

Supervised ML Bundles (cont'd)

- SupportVectorMachines – SVM Bundle (https://github.com/hpcc-systems/SupportVectorMachines)
  - Classification or Regression.
  - Uses the popular LibSVM library under the hood
  - Appropriate for moderate sized datasets or solving many moderate sized problems in parallel.
  - Includes automatic parallelized grid-search to find the best regularization parameters.
- GLM – General Linear Model (https://github.com/hpcc-systems/GLM)
  - Provides various linear methods that can be used when the assumptions of Linear or Logistic Regression don't apply to your data.
  - Includes: binomial, quasibinomial, Poisson, quasipoisson, Gaussian, inverse Gaussian and Gamma GLM families.
  - Can be readily adapted to other error distribution assumptions.
  - Accepts user-defined observation weights/frequencies/priors via a weights argument.
  - Includes analytics.

Supervised ML Bundles (cont'd)

- **LearningTrees** –Random Forests (https://github.com/hpcc-systems/LearningTrees)
  - Classification or Regression
  - One of the best "out-of-box" ML methods as it makes few assumptions about the nature of the data, and is resistant to overfitting. Capable of dealing with large numbers of Independent Variables.
  - Creates a "forest" of diverse Decision Trees and averages the responses of the different Trees.
  - Provides "Feature Importance" metrics.
  - Later versions are planned to support single Decision Trees (C4.5) and Gradient Boosting Trees.
- **Generalized Neural Network (GNN)** – (https://github.com/hpcc-systems/GNN)
  - Combines the parallel processing power of HPCC Systems with the powerful Neural Network capabilities of Keras and Tensorflow
  - Each node is attached to an independent Keras / Tensorflow environment, which can contain various hardware acceleration capabilities such as Graphical Processing Units (GPUs) or Tensor Processing Units (TPUs).
  - Provides a distributed environment that can parallelize all phases of Keras / Tensorflow usage.
  - Provides a Tensor module, allowing users to efficiently encode, decode, and manipulate Tensors within ECL. These Tensor data sets are used as the primary data interface to the GNN functions.

Unsupervised ML Bundles

- **K-Means** – (https://github.com/hpcc-systems/KMeans)
  - Used to automatically cluster unlabeled data
  - It is an implementation of K-Means algorithm, an unsupervised machine learning algorithm to automatically cluster Big Data into different groups.
  - Adopts a hybrid parallelism to enable K-Means calculations on massive datasets and also with a large count of hyper-parameters. The hybrid parallelism can be realized by utilizing the flexible data distribution control and Myriad Interface of HPCC Systems.
- **DBSCAN** – Scalable Parallel DBSCAN Clustering (https://github.com/hpcc-systems/DBSCAN)
  - DBSCAN is Density-Based Spatial Clustering of Applications with Noise
  - An unsupervised machine learning algorithm to automatically cluster the data into subclasses or groups.
  - Provides a range of analytic methods for interpreting the results
- **TextVectors** – ML for Textual Data (https://github.com/hpcc-systems/TextVectors)
  - Allows for the mathematical treatment of textual information.
  - Words, phrases, sentences, and paragraphs can be organized as points.
  - Closeness in space implies closeness in meaning
  - Vectorization allows text to be analyzed numerically and used with other ML techniques.

Machine Learning Tutorial

Learning Trees(a.k.a. Random Forest)

Why Learning Trees?

- Widely considered to be one of the easiest algorithms to use. Why?
  - It makes very little assumption about the data's distribution or its relationships
  - It can handle large numbers of records and large numbers of fields
  - It can handle non-linear and discontinuous relations
  - It almost always works well using the default parameters and without any tuning
- It scales well on HPCC Systems clusters of almost any size
- Its prediction accuracy is competitive with the best state-of-the-art algorithms

Learning Trees Overview

- Random Forests are based on Decision Trees.
- In Decision Trees, you ask a hierarchical series of questions about the data (think a flowchart) until you have asked enough questions to split your data into small groups (leaves) that have a similar result value. Now you can ask the same set of questions of a new data point and return the answer that is representative of the leaf into which is falls.
- Random Forest builds a number of separate and diverse decision trees (a decision forest) and average the results across the forest. The use of many diverse decision trees reduces overfitting since each tree is fit to a different subset of the data, and it will therefore incorporate different "noise" into its model. The aggregation across multiple trees (for the most part) cancels out the noise.

Tutorial Contents

This tutorial example demonstrates the following:

- Installing ML
- Preparing Your Data
- Training the Model
- Assessing the Model
- Making Predictions

Installing ML and LearningTrees

Installing the ML Core Libraries and the LearningTrees Bundles

1. Be sure HPCC Systems Clienttools is installed on your system. Also, you will need **Git for Windows** if you do not already have it installed.
2. Install **HPCC Systems ML_Core**
   From your *clienttools/bin* folder, run in the Command Window:

   ecl bundle install https://github.com/hpcc-systems/ML_Core.git
3. Install the **HPCC Systems LearningTrees** bundle. Run:

   ecl bundle install https://github.com/hpcc-systems/LearningTrees.git

NOTE: For PC users, the ecl bundle install must be run as an Administrator. Right click on the command icon and select "Run as administrator" when you start your command window.

Easy! Now let's get your data ready to use.

Preparing Your Data

The data should contain *all numeric values*, and the *first field* of your record is an UNSIGNED unique identifier.

1. The first step is to segregate your data into a *training* set and a *testing* set, using a random sample.

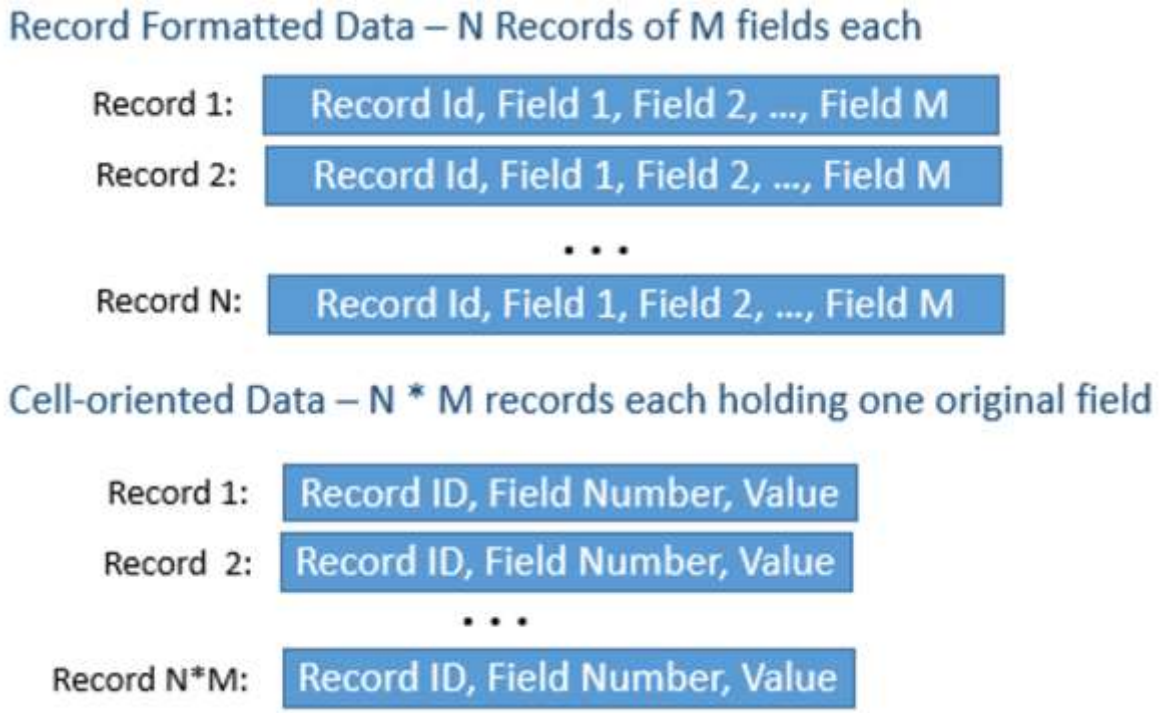
Download and import MLTutorial.MOD

Open **MLTutorial.Prep01.ECL**

Preparing Your Data

2. Convert your data to the form used by the ML bundles. ML requires the data in a cell-oriented matrix layout known as the **NumericField**.

```
IMPORT ML_Core;
ML_Core.ToField(myTrainData, myTrainDataNF);
```



Open **MLTutorial.Convert02.ECL**

At this point, *myTrainDataNF* will contain your converted training data. Note that the MACRO does not return the converted data but creates an *new* in-line definition that contains it.

Preparing Your Data (More)

3. Separate the Independent fields from the Dependent fields:

```
myIndTrainDataNF := myTrainDataNF(number < 11); // Number is the field number

myDepTrainDataNF := PROJECT(myTrainDataNF(number = 11),
                            TRANSFORM(RECORDOF(LEFT), SELF.number := 1, SELF := LEFT));
```

Note that using the PROJECT to set the number field to 1 is not strictly necessary, but is a good practice. This indicates that it is the first field of the dependent data. Since there is only one Dependent field, we number it accordingly.

4. Do the same thing for the test data:

```
myIndTestDataNF := myTestDataNF(number < 11); // Number is the field number

myDepTestDataNF := PROJECT(myTestDataNF(number = 11),
                           TRANSFORM(RECORDOF(LEFT), SELF.number := 1, SELF := LEFT));
```

We are now ready to train the model and assess our results!

Training the Model

1. Define our learner (Regression/Classification Forest)
2. Train and Retrieve the Model
   - Regression (**MLTutorial.Train03**)
     - ✓RegressionForest
     - ✓GetModel/Predict
     - ✓Accuracy Assessment
   - Classification (**MLTutorial.Train04**)
     - ✓Discretize (ByRounding)
     - ✓ClassificationForest
     - ✓GetModel/Classify
     - ✓Accuracy Assessment

Assessing the Model

Assessing the Model is fairly easy:

1. Create a new set of Independent field samples
2. Use the Predict FUNCTION for Regression:

```
predictedValues := myLearnerR.Predict(myModelR, myNewIndData);
```

3. Use the Classify FUNCTION for Classification:

```
predictedClasses := myLearnerC.Classify(myModelC, myNewIndData);
```

Note: For best results, your train/test data should be representative of the population whose values you will try to predict.

Conclusion

- You now know everything you need to know to build and test ML models against your data and to use those models to predict qualitative or quantitative values (i.e. classification or regression).
- If you want to use a different ML bundle, you'll find that all the bundles operate in a very similar fashion, with minor variation.
- We've utilized only the most basic aspects of the ML bundles.

And finally:

ML's conceptual simplicity is somewhat misleading. Each algorithm has its own quirks, (assumptions and constraints) that need to be taken into account in order to maximize the predictive accuracy.

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