# Low-code development of computation applications

## Baltic Large-Scale Computing Platform

Michał Śmiałek
Warsaw University of Technology

# Introduction

- Computation systems
  - **High Performance Computing system** – a big, centralized supercomputer (typically: homogeneous system), dedicated software
  - **Large Scale Computing system** – a network of many interconnected computation clusters (typically: heterogeneous), based on standard orchestration solutions
- Low-code software development
  - High-level, graphical programming language – programming = building visual models
  - Model-Driven Web Engineering – development of web applications using visual models
  - In LSC – creating computation applications visually, through a web development interface

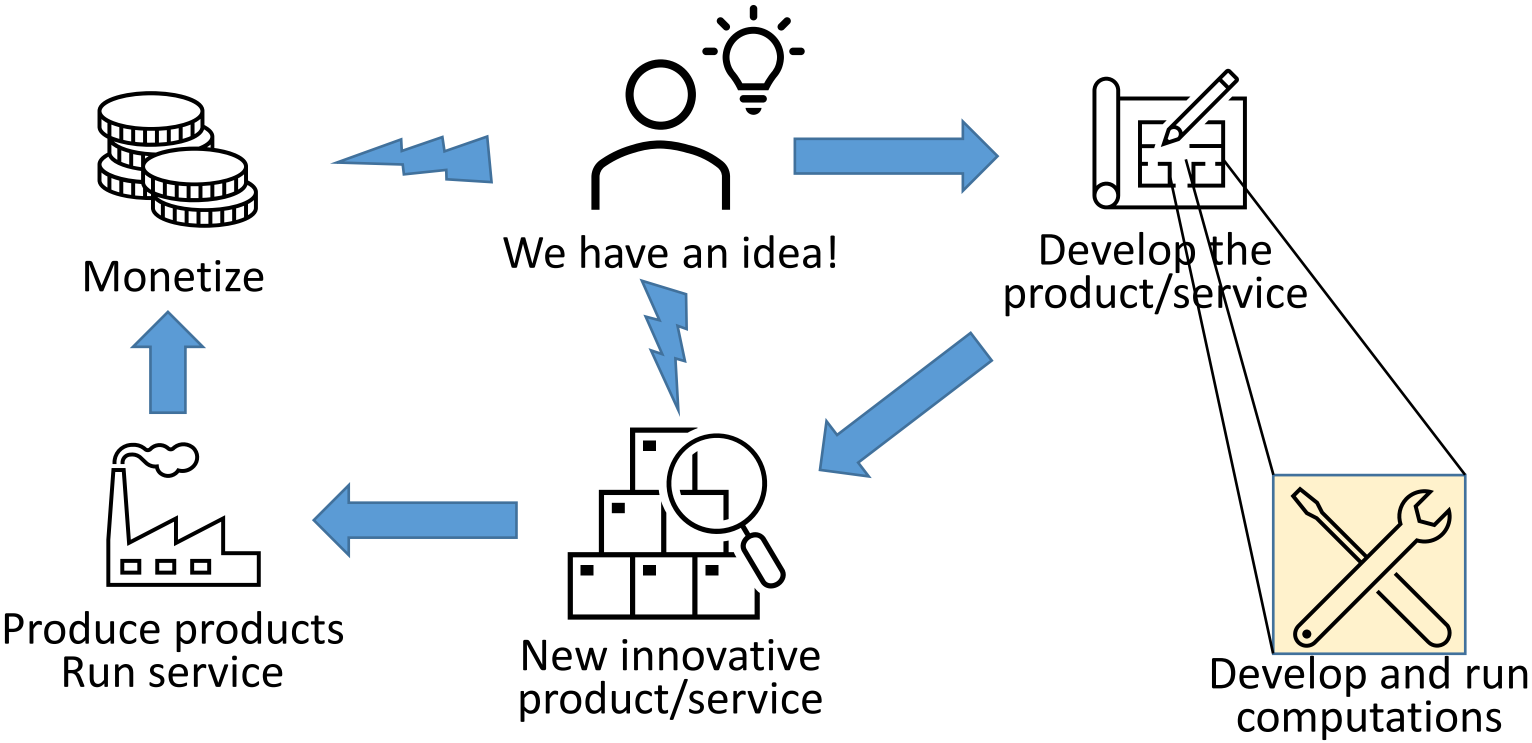
# Agenda

- What is the problem?
- How to make LSC easy to use?
- Visual computation language
- Runtime environment for distributed computations
- Development of computation modules

Interreg
Baltic Sea Region
EUROPEAN UNION
EUROPEAN REGIONAL DEVELOPMENT FUND

# :: What is the problem? ::

# Turning great ideas into ready solutions



We have an idea!

Develop the product/service

Develop and run computations

New innovative product/service

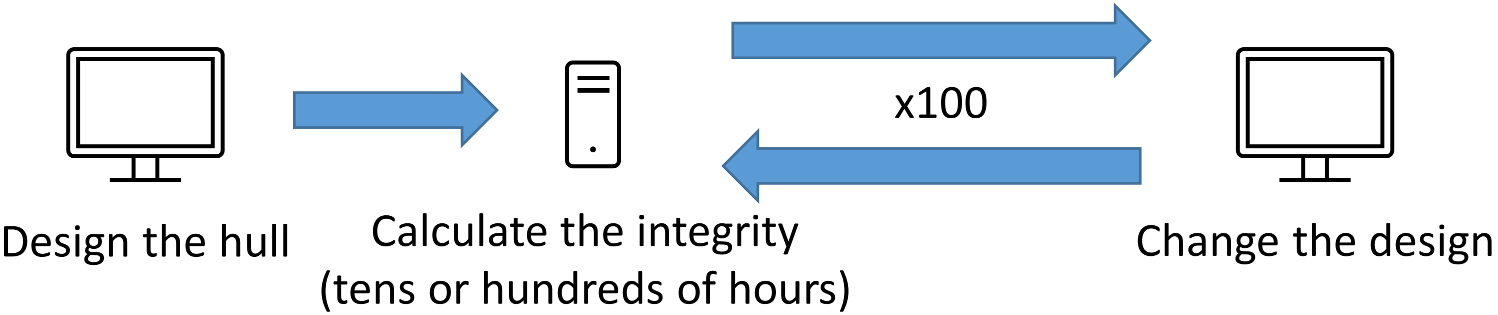
Produce products
Run service

Monetize

P: New Catamaran Yacht
P: New Virus Vaccine
S: New Airport Security System
S: New Weather Forecasting Service

Calculate yacht's hull aerodynamics…
Compute DNA sequences of a virus…
Train a neural network (AI system)…
Test weather forecast algorithms…

# Ship’s hull optimisation



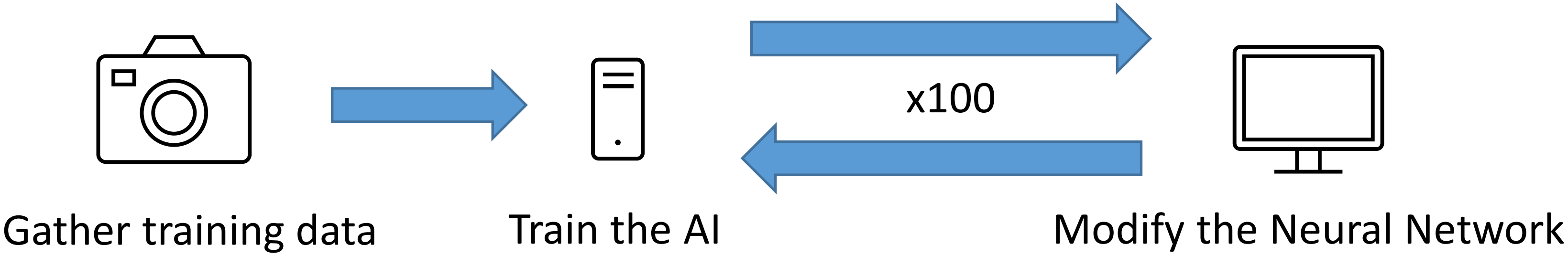
Design the hull

Calculate the integrity
(tens or hundreds of hours)

x100

Change the design

# AI welding quality inspection



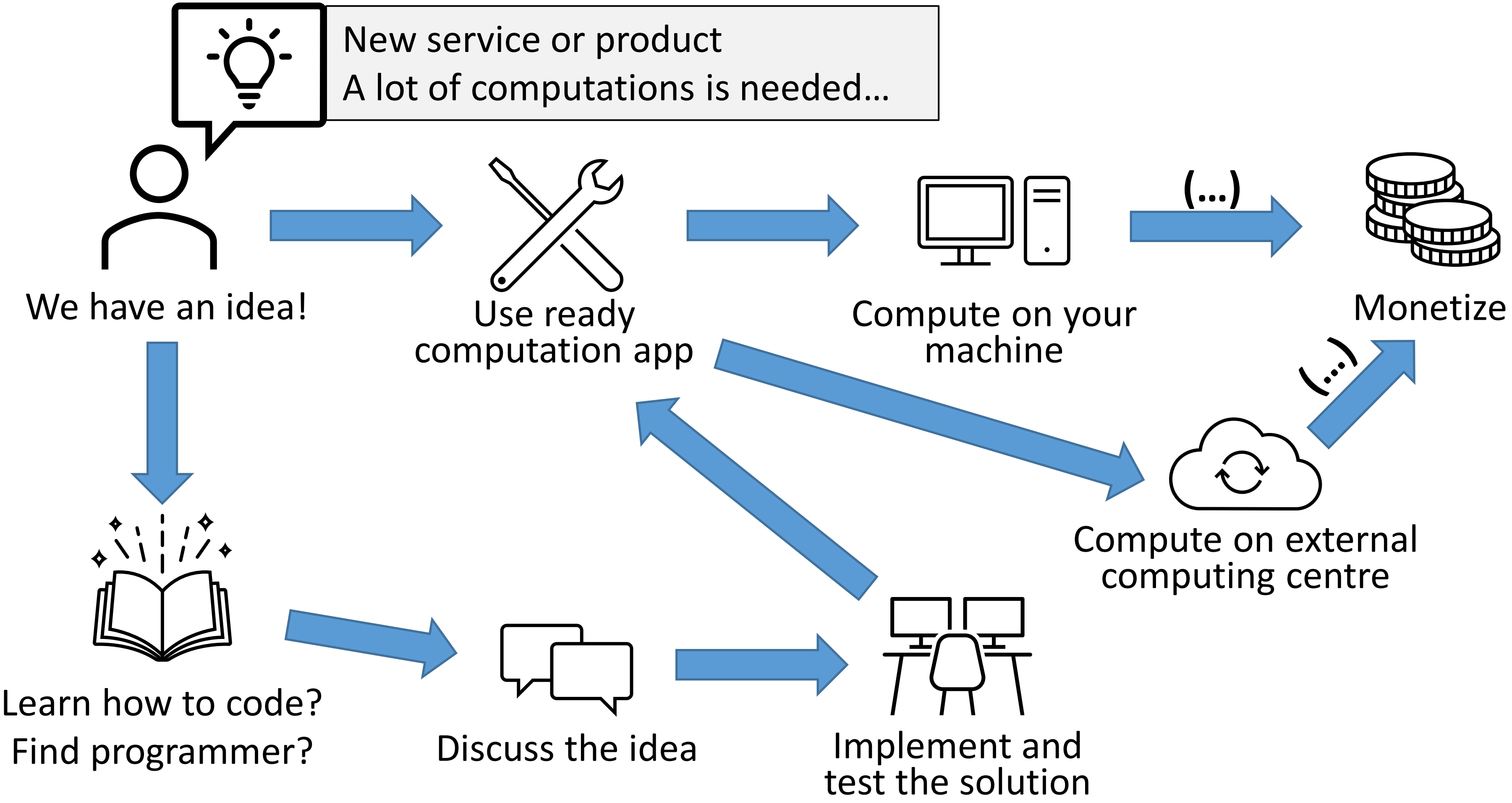
x100

Gather training data

Train the AI

Modify the Neural Network

# How to develop an innovation?



New service or product

A lot of computations is needed…

We have an idea!

Use ready computation app

Compute on your machine

(…)

Monetize

(…)

Compute on external computing centre

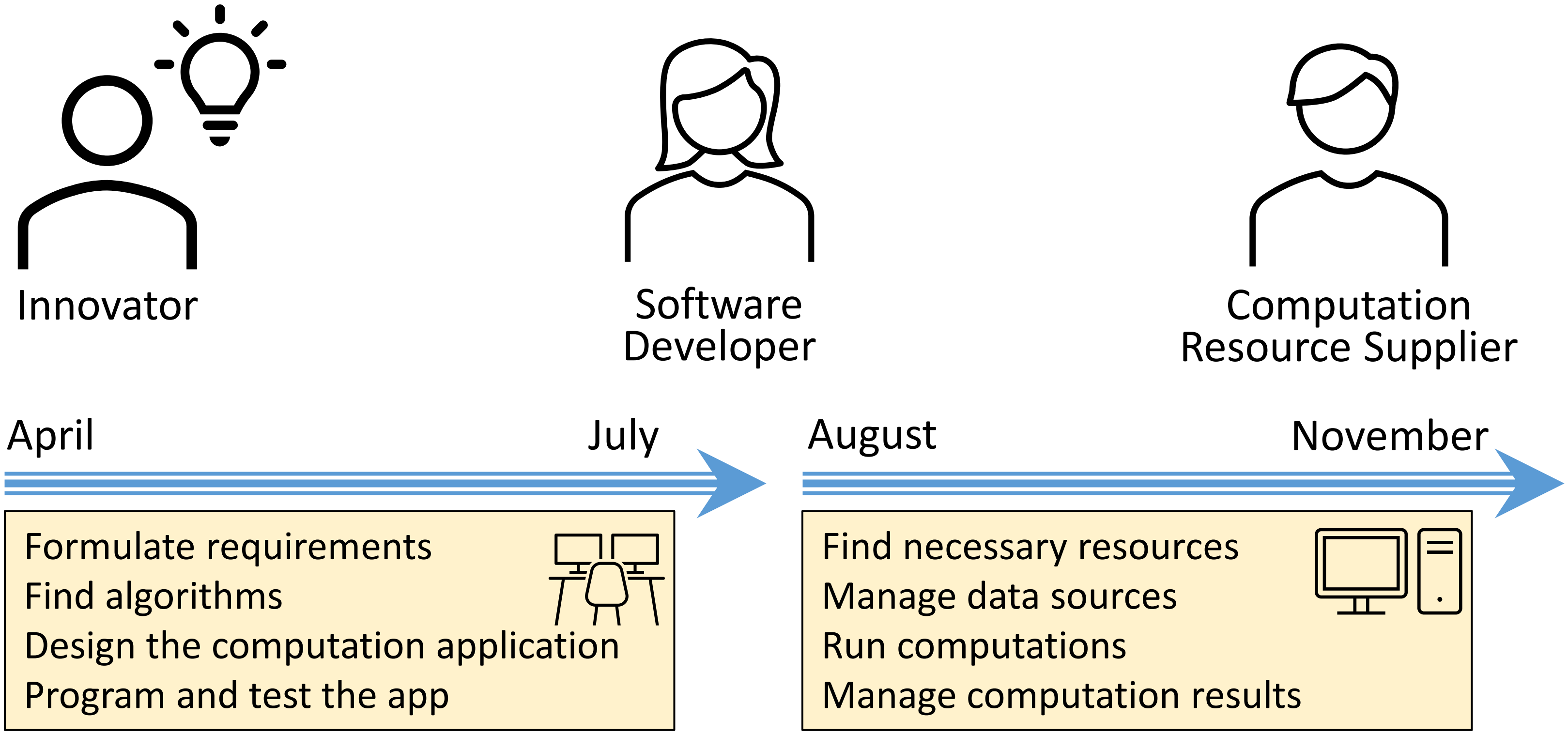
Learn how to code?

Find programmer?

Discuss the idea

Implement and test the solution

# How long does it take?



Innovator

Software Developer

Computation Resource Supplier

April → July

- Formulate requirements
- Find algorithms
- Design the computation application
- Program and test the app

August → November

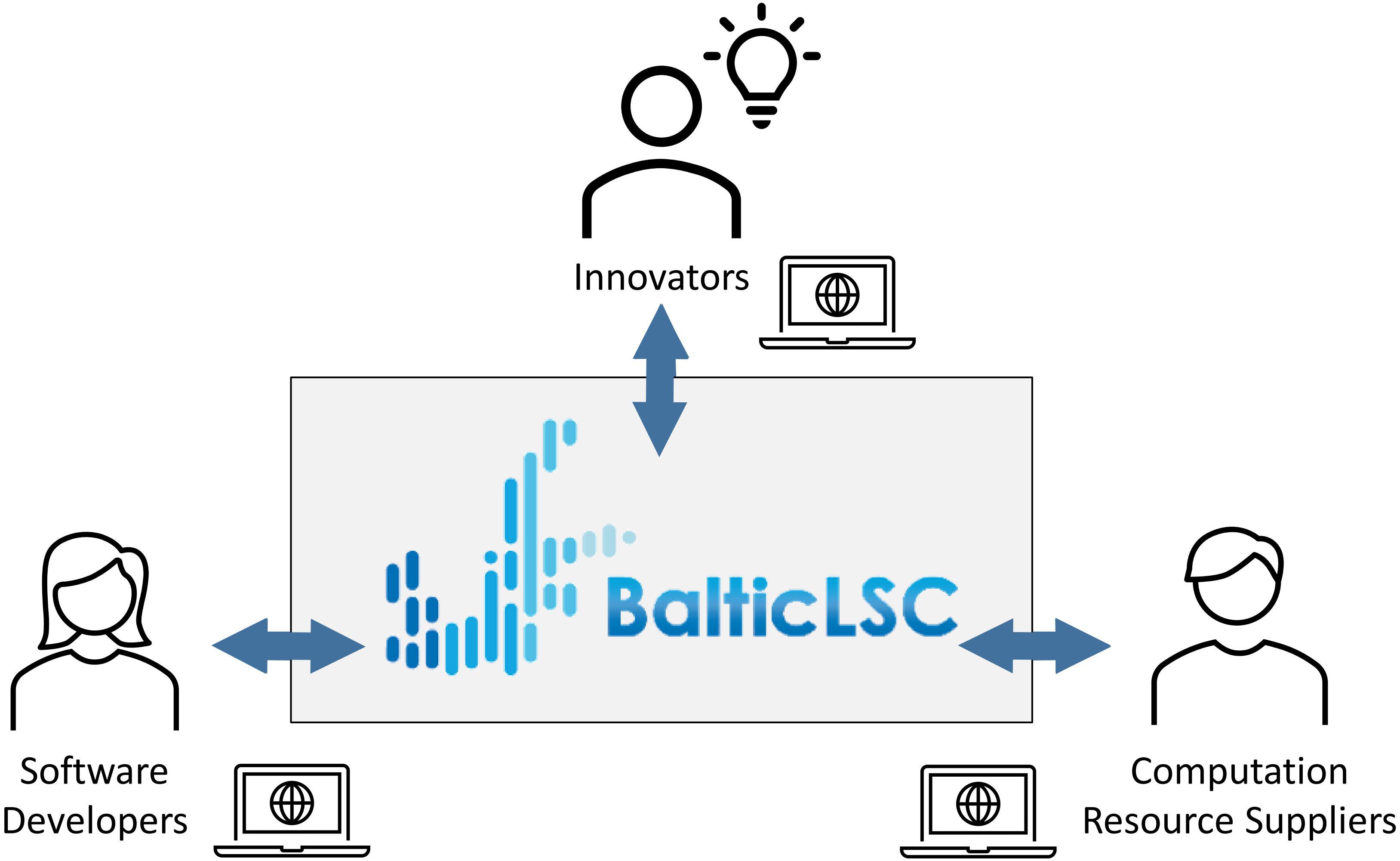
- Find necessary resources
- Manage data sources
- Run computations
- Manage computation results

BalticLSC

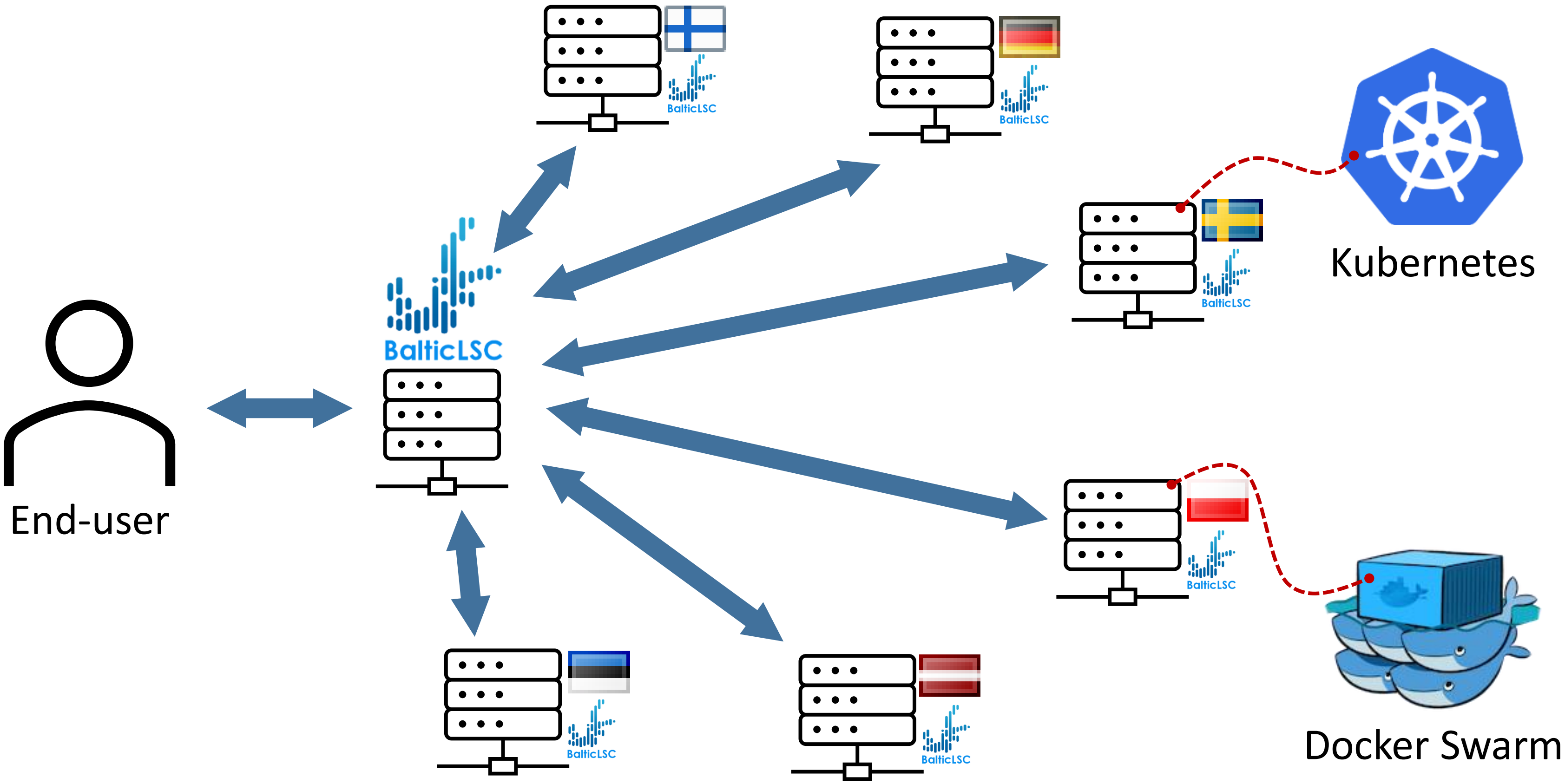
**Goal: shorten this significantly**

# :: How to make LSC easy to use? ::

# One place to connect everyone

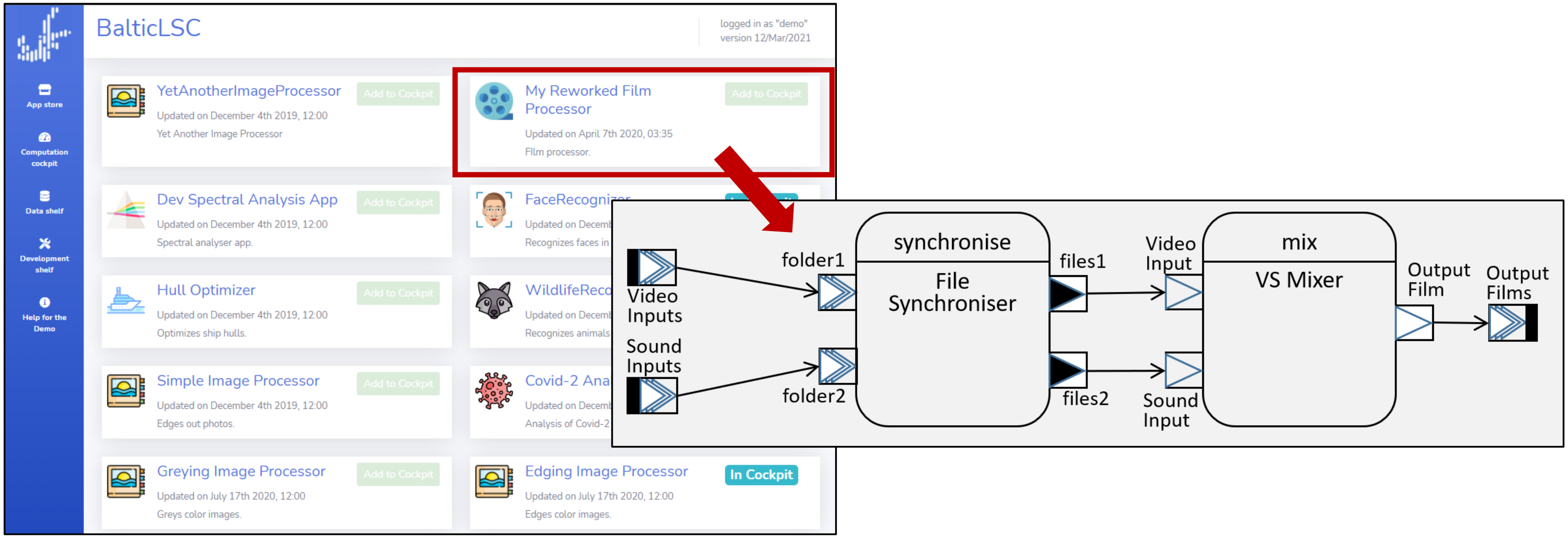


# BalticLSC Computation Platform



**Computation Clusters** of various size and operating software can participate in the BalticLSC Computation Platform

# Select or easily develop your computation apps



BalticLSC

logged in as "demo"
version 12/Mar/2021

App store
Computation cockpit
Data shelf
Development shelf
Help for the Demo

YetAnotherImageProcessor — Add to Cockpit
Updated on December 4th 2019, 12:00
Yet Another Image Processor

My Reworked Film Processor — Add to Cockpit
Updated on April 7th 2020, 03:35
Film processor.

Dev Spectral Analysis App — Add to Cockpit
Updated on December 4th 2019, 12:00
Spectral analyser app.

FaceRecogni
Updated on Decem
Recognizes faces in

Hull Optimizer — Add to Cockpit
Updated on December 4th 2019, 12:00
Optimizes ship hulls.

WildlifeReco
Updated on Decem
Recognizes animals

Simple Image Processor — Add to Cockpit
Updated on December 4th 2019, 12:00
Edges out photos.

Covid-2 Ana
Updated on Decem
Analysis of Covid-2

Greying Image Processor — Add to Cockpit
Updated on July 17th 2020, 12:00
Greys color images.

Edging Image Processor — In Cockpit
Updated on July 17th 2020, 12:00
Edges color images.

- Select a ready app from an **App Store**
- Or: select ready computation modules
- Compose your application from modules
- Use easy-to-understand graphical Computation Application Language (**CAL**)

# Components of the BalticLSC Platform

Application Store

Computation Cockpit

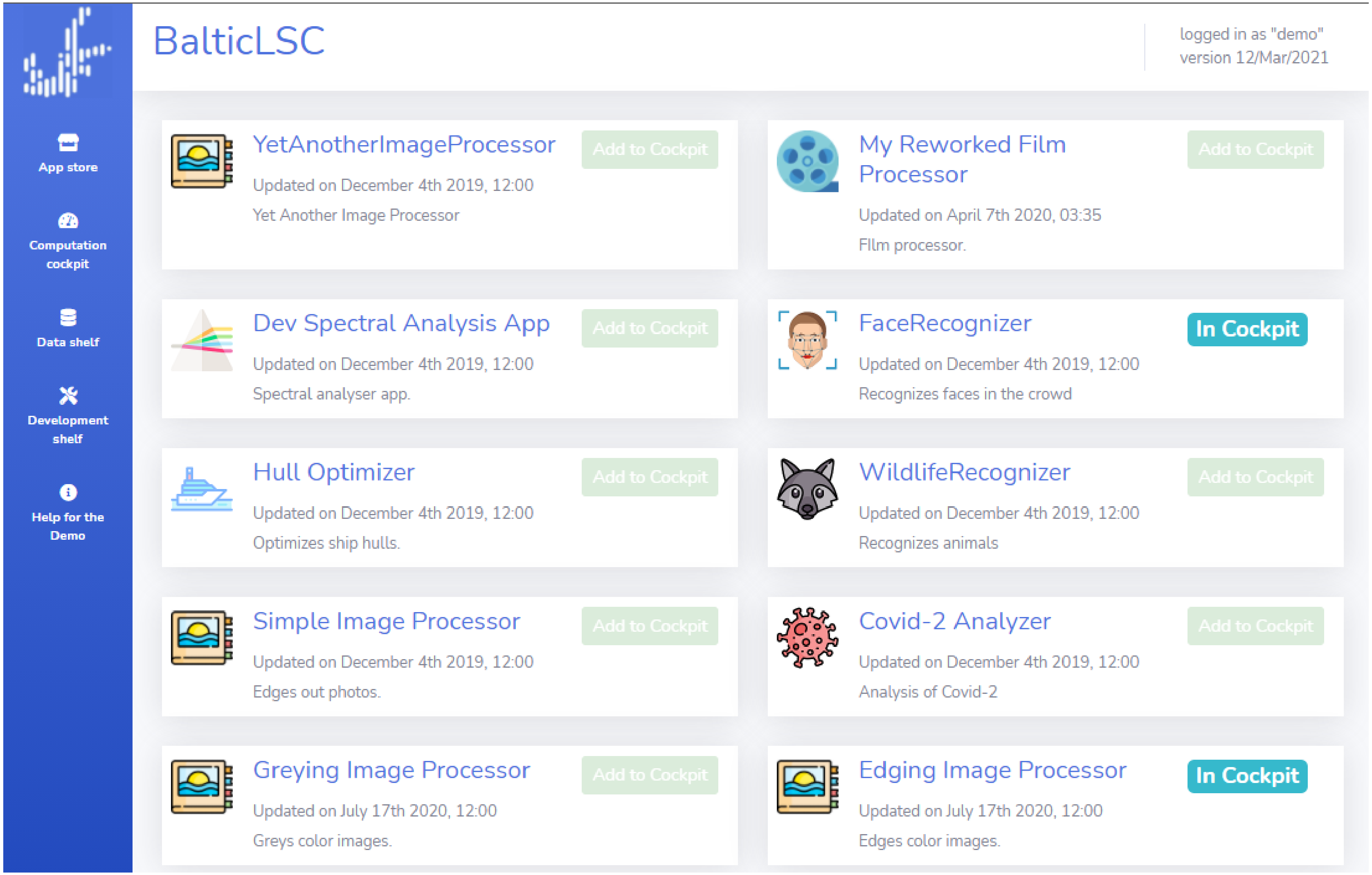
Development Shelf

BalticLSC

Data Shelf

CAL Low-Code Editor

# Application Store



BalticLSC

logged in as "demo"
version 12/Mar/2021

**YetAnotherImageProcessor** — Add to Cockpit

Updated on December 4th 2019, 12:00

Yet Another Image Processor

**My Reworked Film Processor** — Add to Cockpit

Updated on April 7th 2020, 03:35

Film processor.

**Dev Spectral Analysis App** — Add to Cockpit

Updated on December 4th 2019, 12:00

Spectral analyser app.

**FaceRecognizer** — In Cockpit

Updated on December 4th 2019, 12:00

Recognizes faces in the crowd

**Hull Optimizer** — Add to Cockpit

Updated on December 4th 2019, 12:00

Optimizes ship hulls.

**WildlifeRecognizer** — Add to Cockpit

Updated on December 4th 2019, 12:00

Recognizes animals

**Simple Image Processor** — Add to Cockpit

Updated on December 4th 2019, 12:00

Edges out photos.

**Covid-2 Analyzer** — Add to Cockpit

Updated on December 4th 2019, 12:00

Analysis of Covid-2

**Greying Image Processor** — Add to Cockpit

Updated on July 17th 2020, 12:00

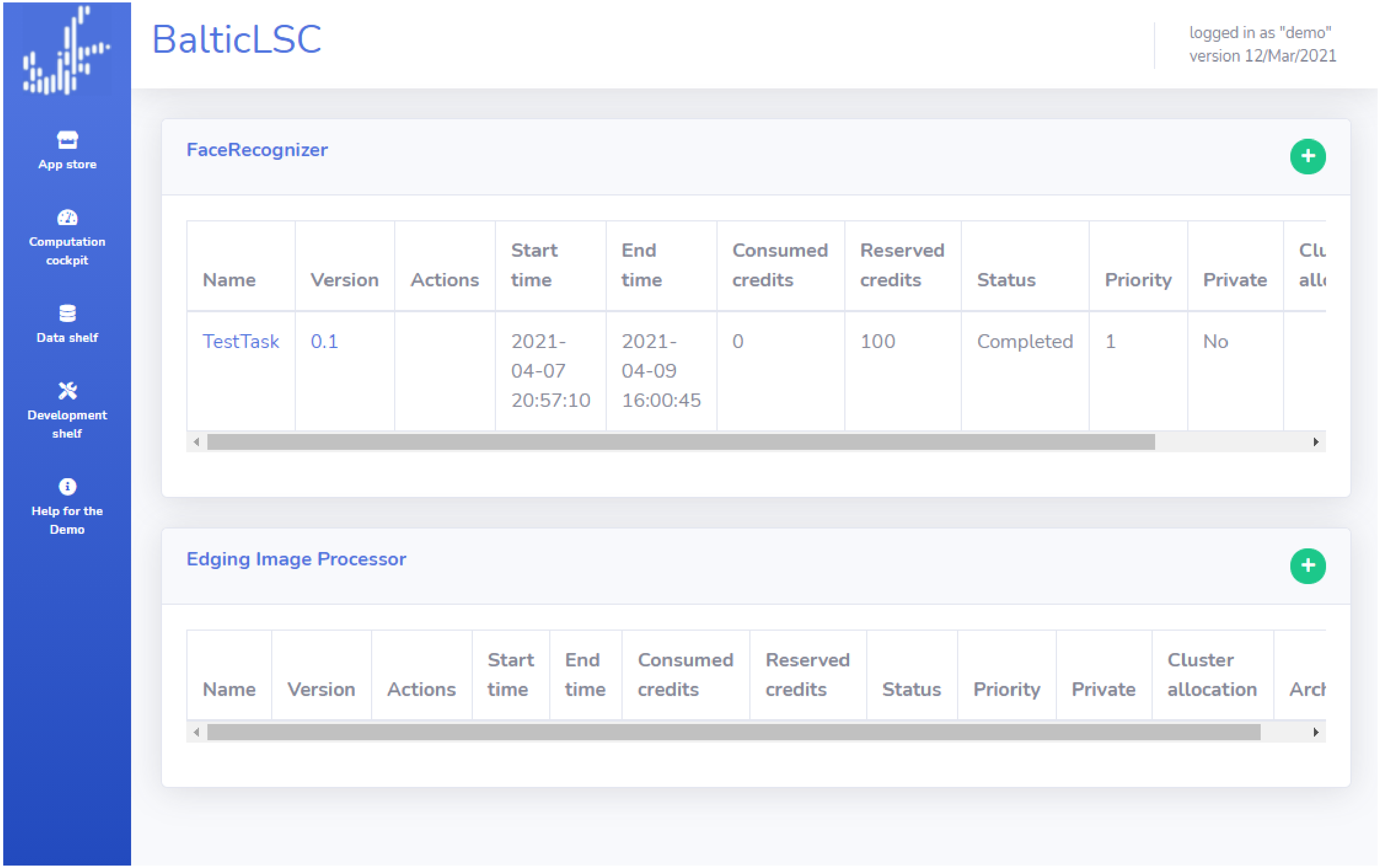
Greys color images.

**Edging Image Processor** — In Cockpit

Updated on July 17th 2020, 12:00

Edges color images.

# Computation Cockpit



BalticLSC

logged in as "demo"
version 12/Mar/2021

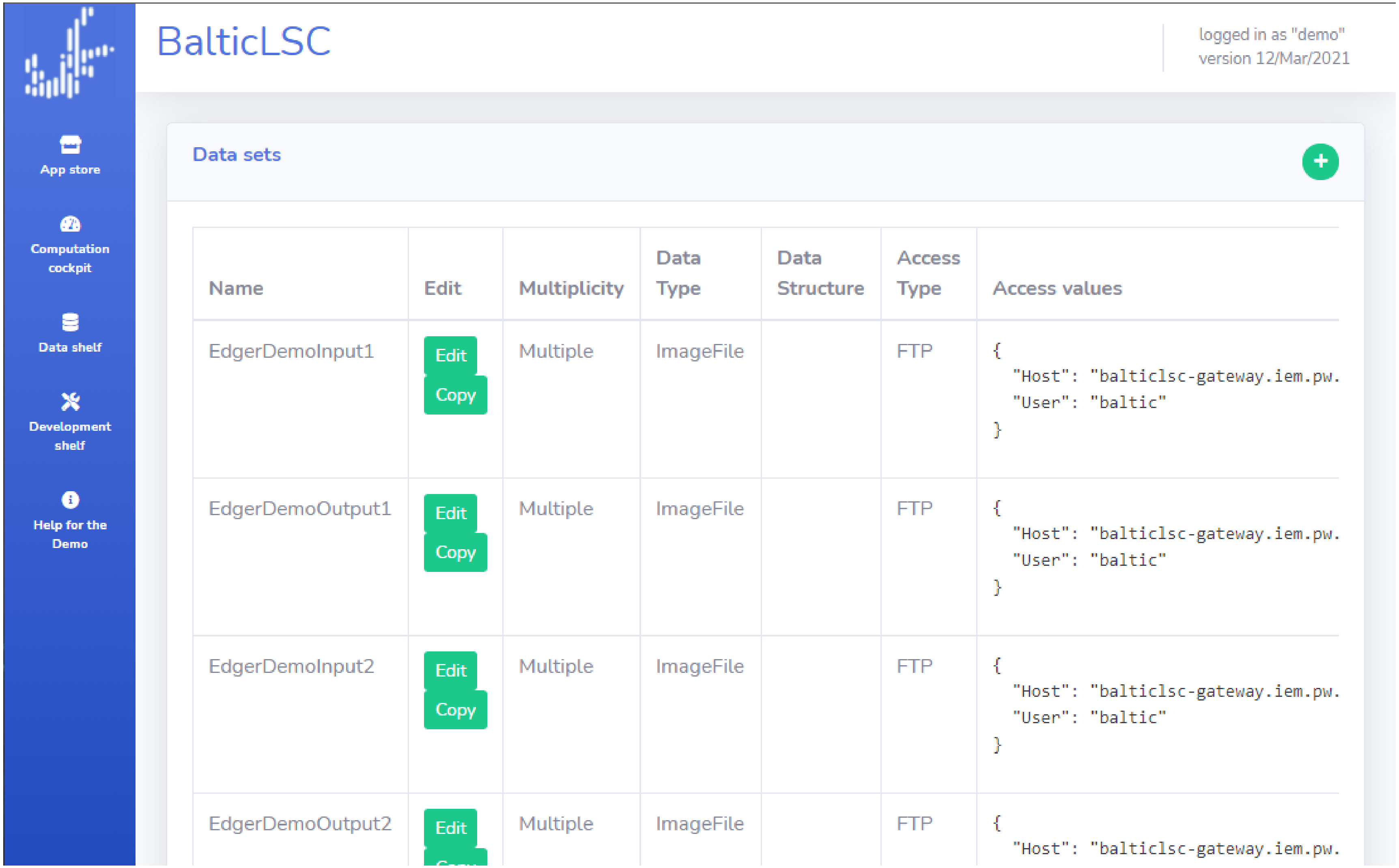
FaceRecognizer

| Name | Version | Actions | Start time | End time | Consumed credits | Reserved credits | Status | Priority | Private | Clu all |
|---|---|---|---|---|---|---|---|---|---|---|
| TestTask | 0.1 | | 2021-04-07 20:57:10 | 2021-04-09 16:00:45 | 0 | 100 | Completed | 1 | No | |

Edging Image Processor

| Name | Version | Actions | Start time | End time | Consumed credits | Reserved credits | Status | Priority | Private | Cluster allocation | Arc |
|---|---|---|---|---|---|---|---|---|---|---|---|

# Data Shelf



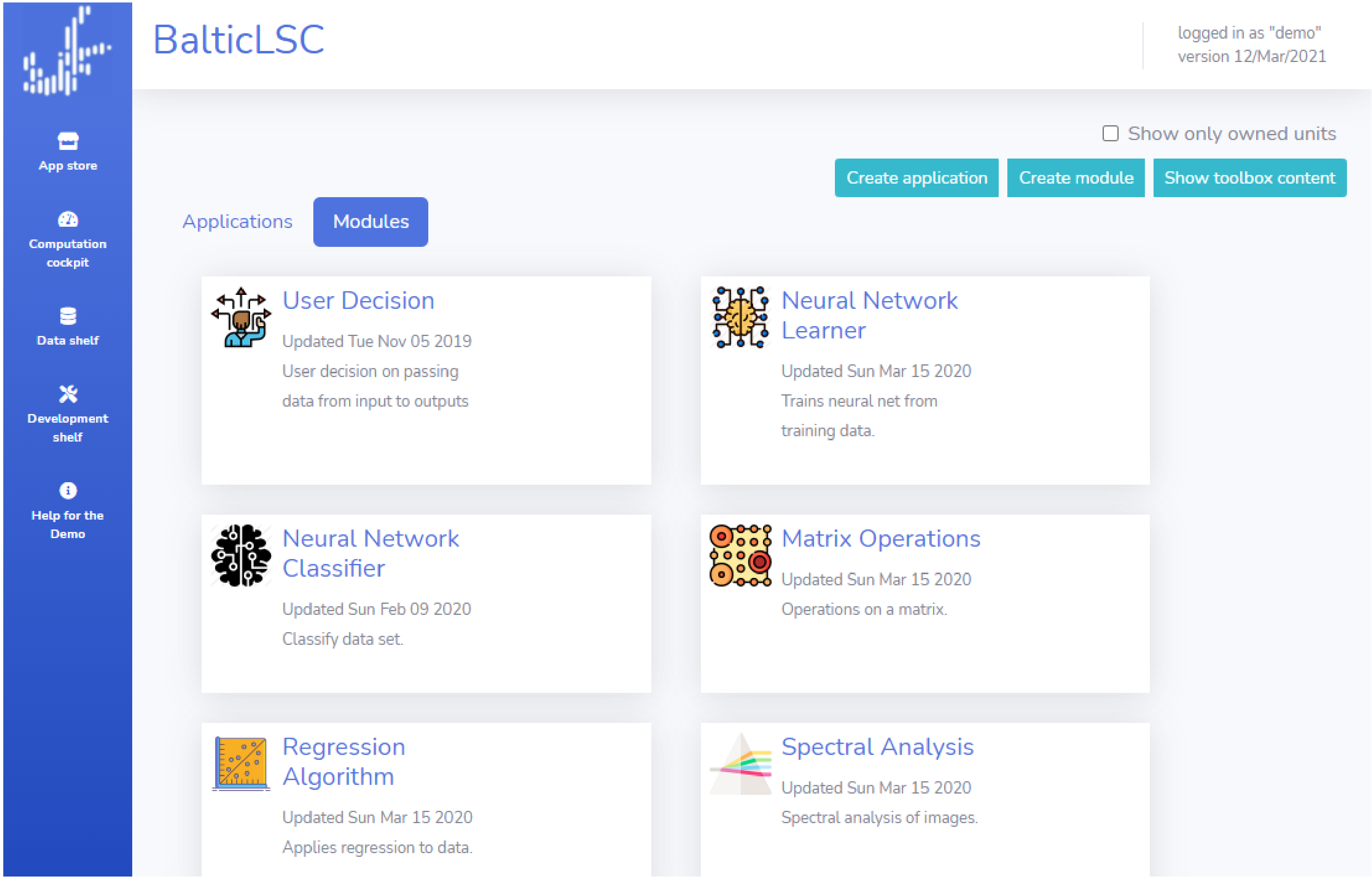
BalticLSC

logged in as "demo"
version 12/Mar/2021

Data sets

| Name | Edit | Multiplicity | Data Type | Data Structure | Access Type | Access values |
|---|---|---|---|---|---|---|
| EdgerDemoInput1 | Edit<br>Copy | Multiple | ImageFile |  | FTP | {<br>"Host": "balticlsc-gateway.iem.pw.<br>"User": "baltic"<br>} |
| EdgerDemoOutput1 | Edit<br>Copy | Multiple | ImageFile |  | FTP | {<br>"Host": "balticlsc-gateway.iem.pw.<br>"User": "baltic"<br>} |
| EdgerDemoInput2 | Edit<br>Copy | Multiple | ImageFile |  | FTP | {<br>"Host": "balticlsc-gateway.iem.pw.<br>"User": "baltic"<br>} |
| EdgerDemoOutput2 | Edit | Multiple | ImageFile |  | FTP | {<br>"Host": "balticlsc-gateway.iem.pw. |

# Development Shelf



BalticLSC

logged in as "demo"
version 12/Mar/2021

□ Show only owned units

**User Decision**
Updated Tue Nov 05 2019
User decision on passing data from input to outputs

**Neural Network Learner**
Updated Sun Mar 15 2020
Trains neural net from training data.

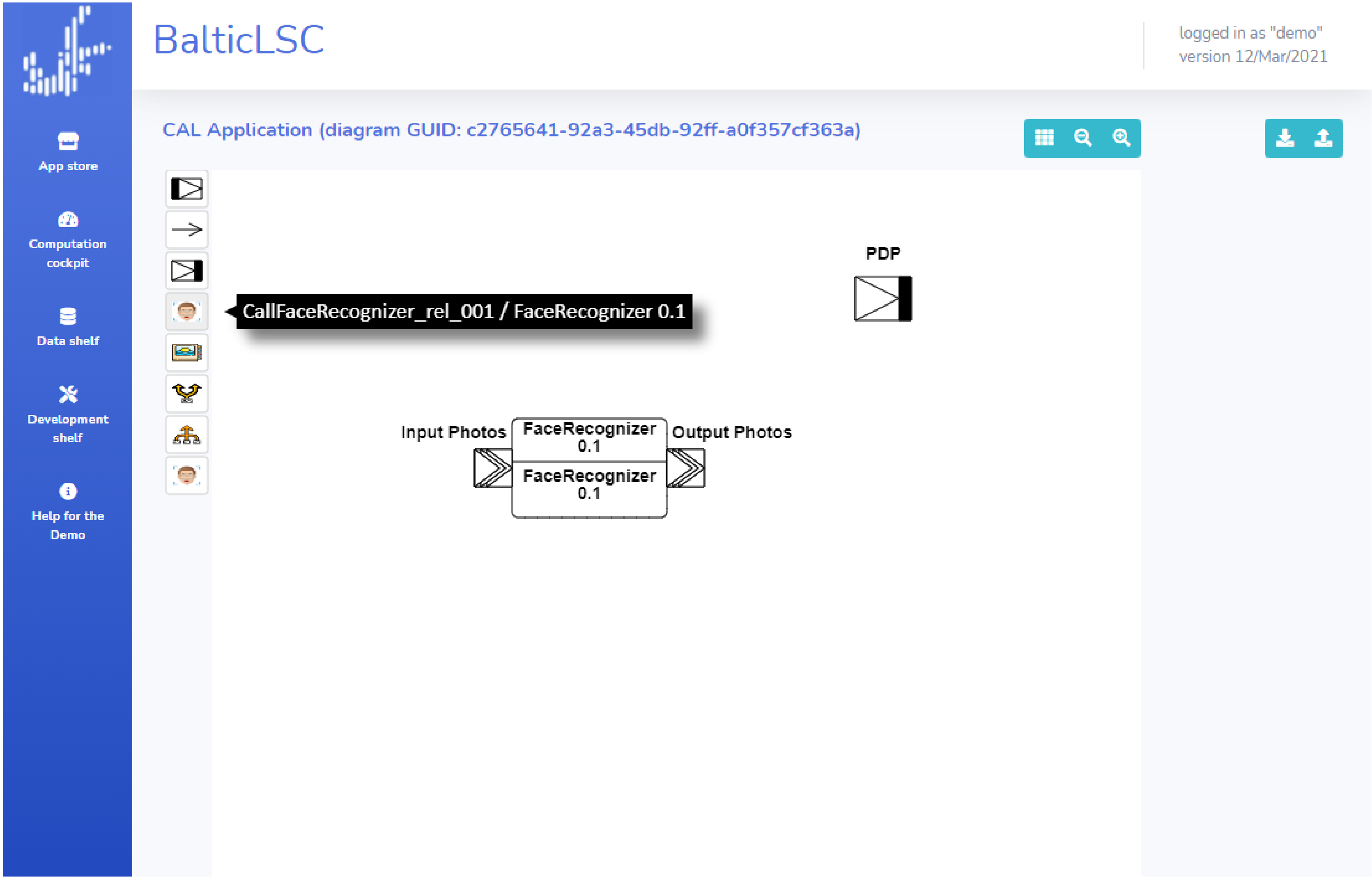
**Neural Network Classifier**
Updated Sun Feb 09 2020
Classify data set.

**Matrix Operations**
Updated Sun Mar 15 2020
Operations on a matrix.

**Regression Algorithm**
Updated Sun Mar 15 2020
Applies regression to data.

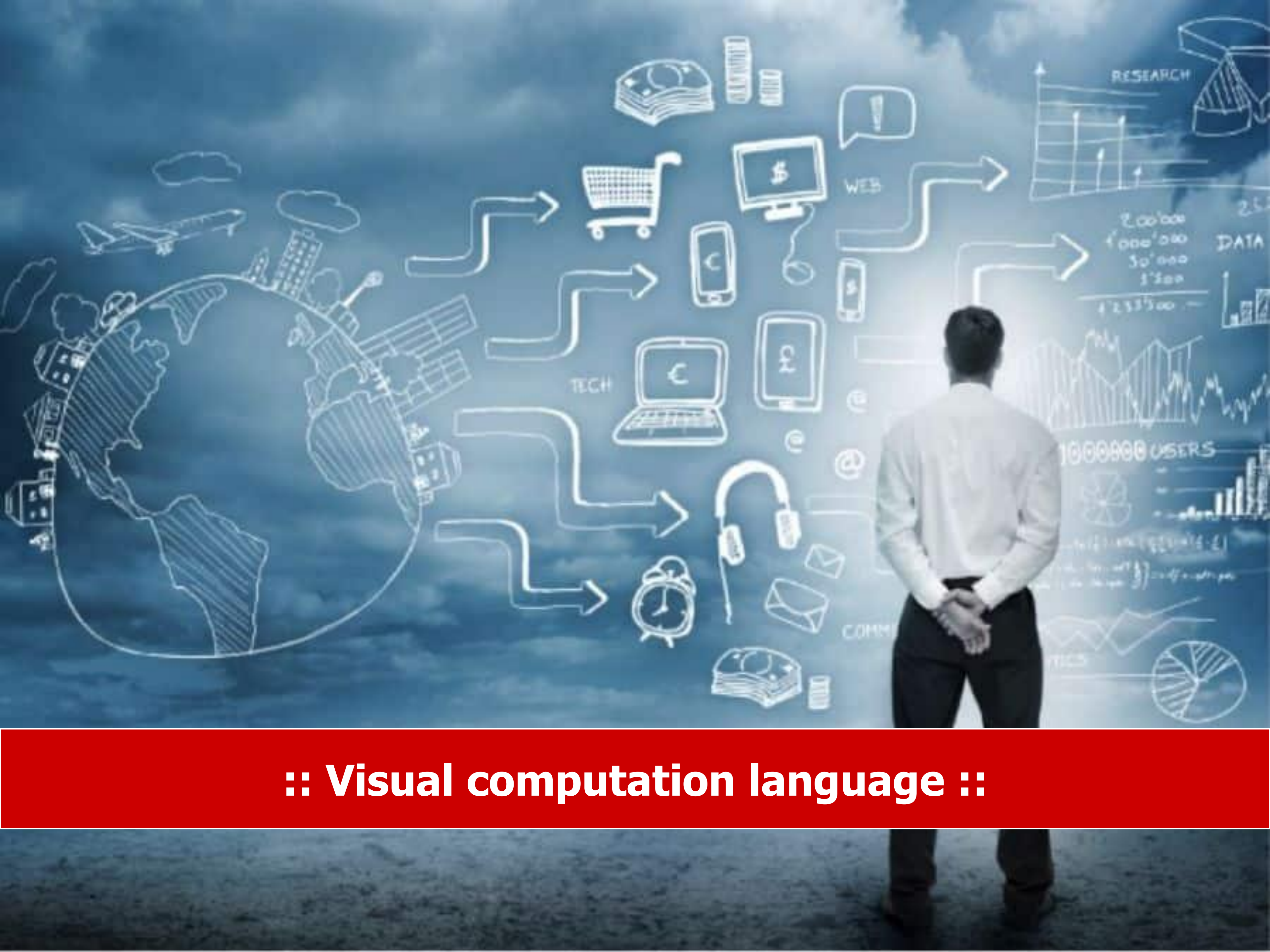
**Spectral Analysis**
Updated Sun Mar 15 2020
Spectral analysis of images.

# CAL Editor



CAL Application (diagram GUID: c2765641-92a3-45db-92ff-a0f357cf363a)

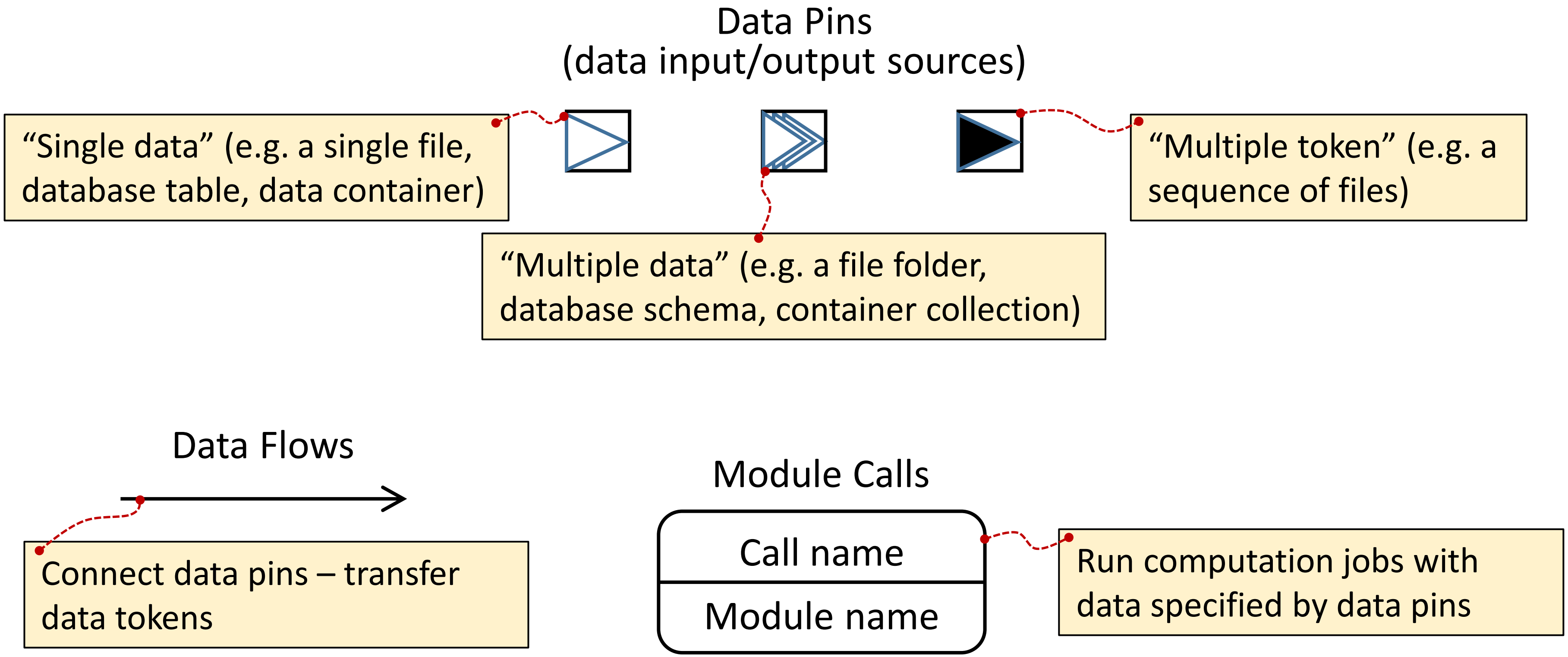
CallFaceRecognizer_rel_001 / FaceRecognizer 0.1



# :: Visual computation language ::

# BalticLSC Applications

- Computation Application Language – a visual (graphical) low-code language



Data Pins
(data input/output sources)

“Single data” (e.g. a single file, database table, data container)

“Multiple data” (e.g. a file folder, database schema, container collection)

“Multiple token” (e.g. a sequence of files)

Data Flows

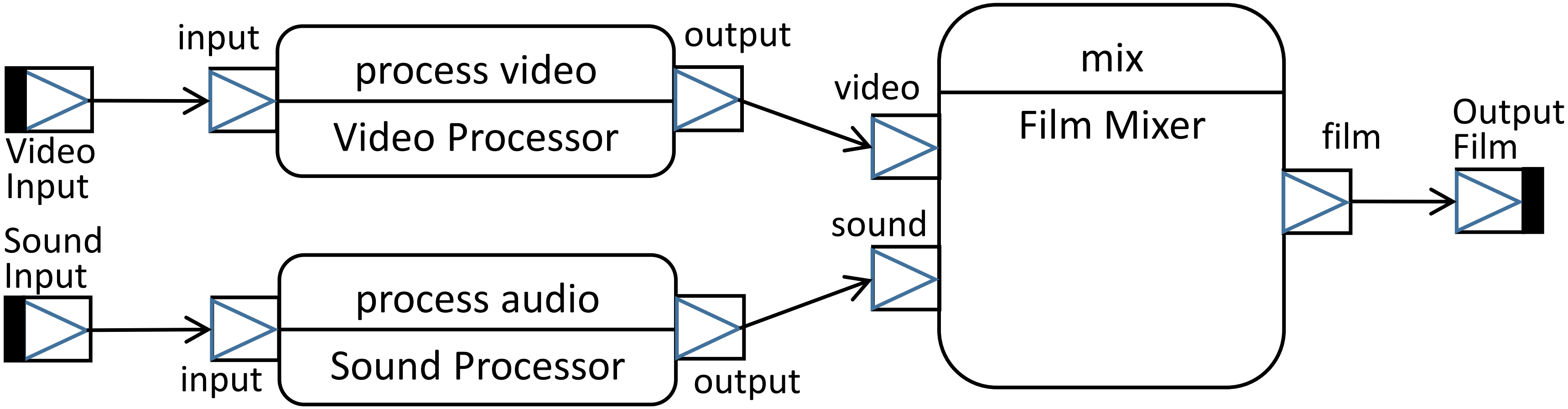
Connect data pins – transfer data tokens

Module Calls

Run computation jobs with data specified by data pins

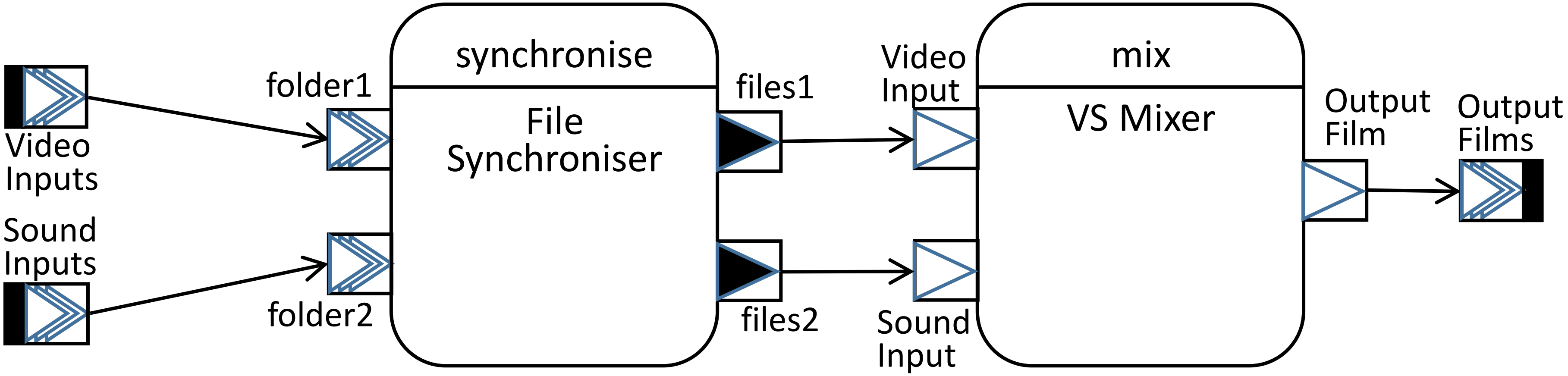
# CAL Example – VS Mixer

- Process video and audio, and then mix them into a single film
- Three module calls – three “Job” executions



# CAL Example – Multi VS Mixer

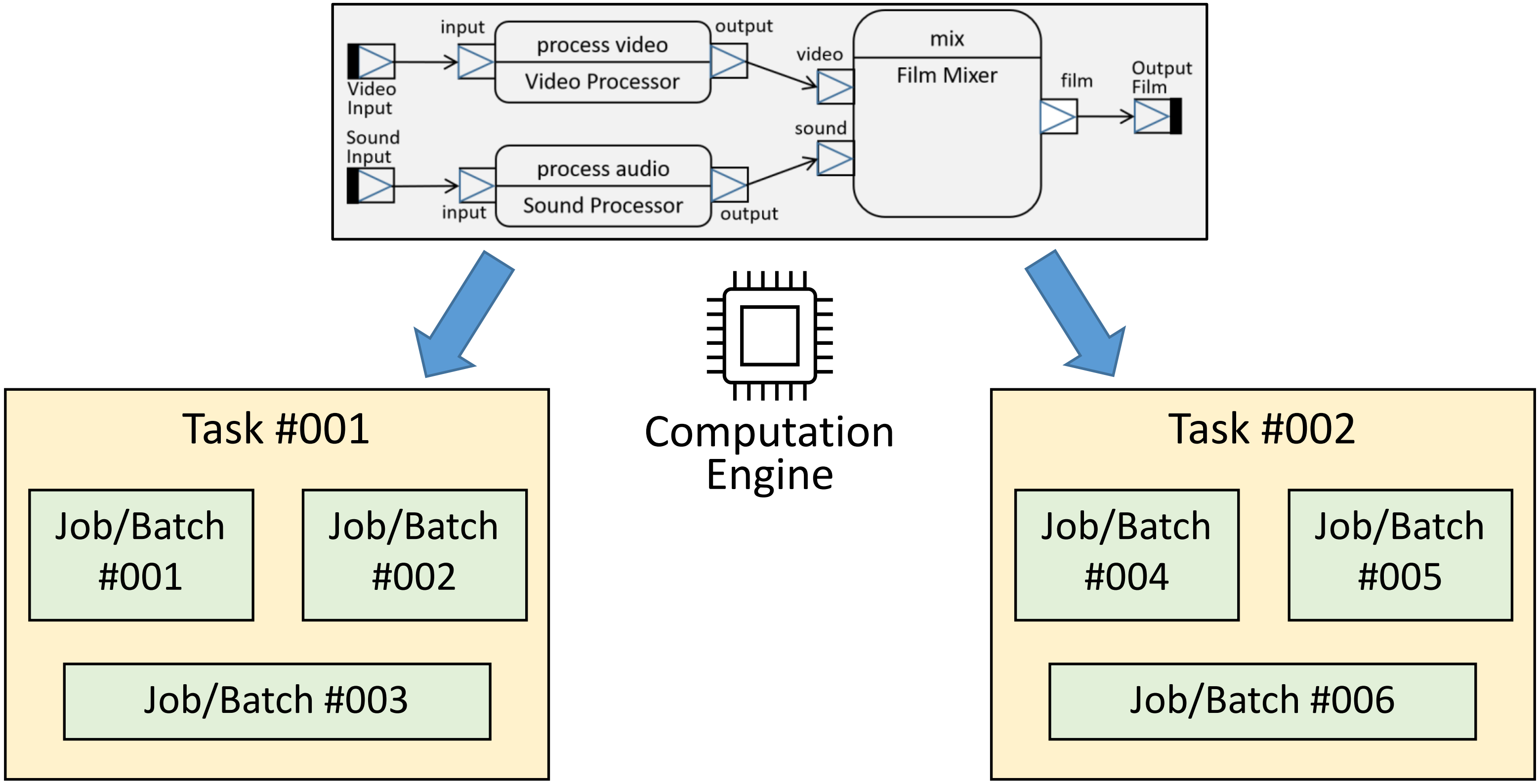
- Process many video-audio pairs
- Calls VS Mixer
- Four module calls (3+1) – many “Job” executions



# :: Runtime environment for distributed computations ::

# Running CAL Applications

- Application → Task
- Module Calls → Jobs (Batches)



Computation Engine

Task #001

Job/Batch #001

Job/Batch #002

Job/Batch #003

Task #002

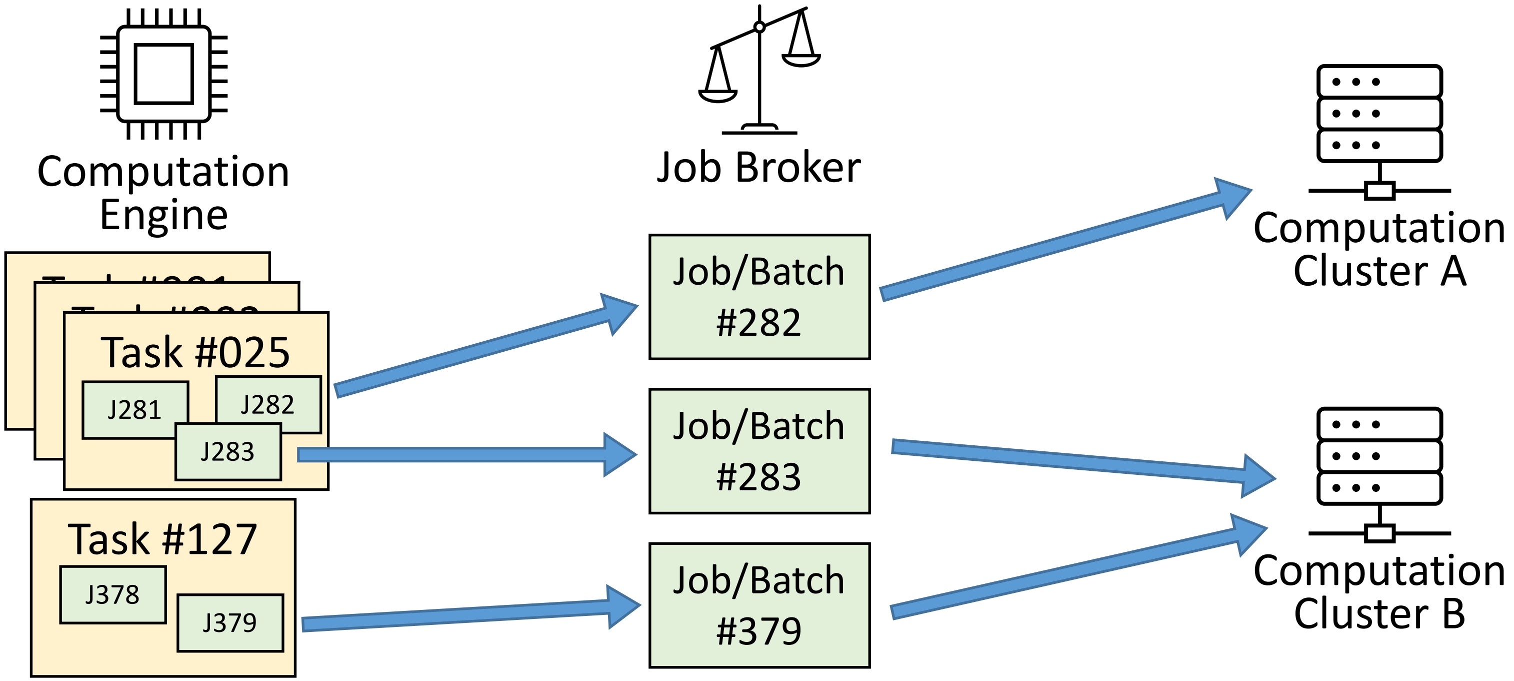
Job/Batch #004

Job/Batch #005

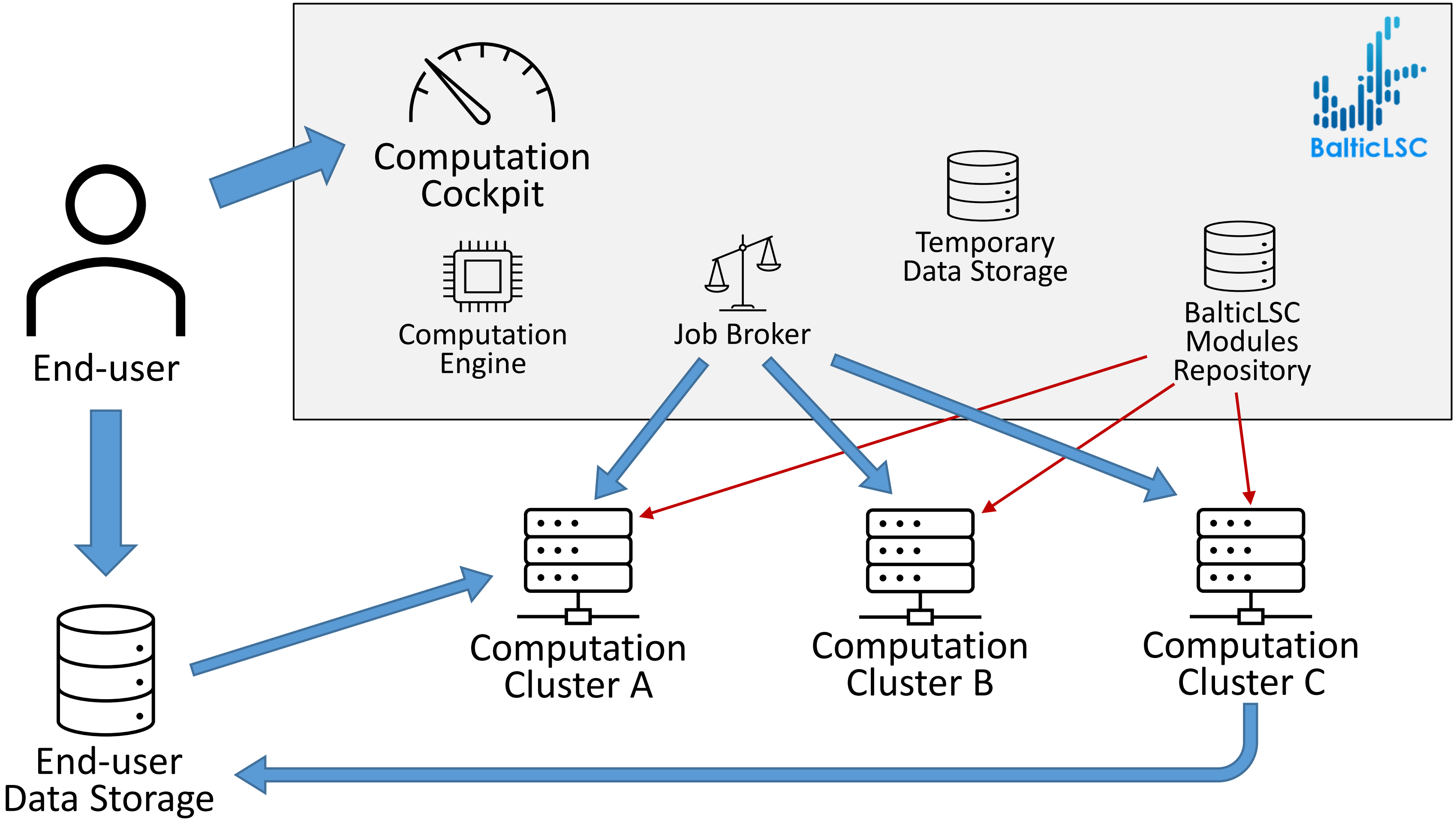
Job/Batch #006

# Assigning Jobs to Computation Clusters

- Computation Engine makes sure that multiple tasks/jobs from different users can run in parallel
- Job Broker makes sure that clusters are assigned according to job's required resources

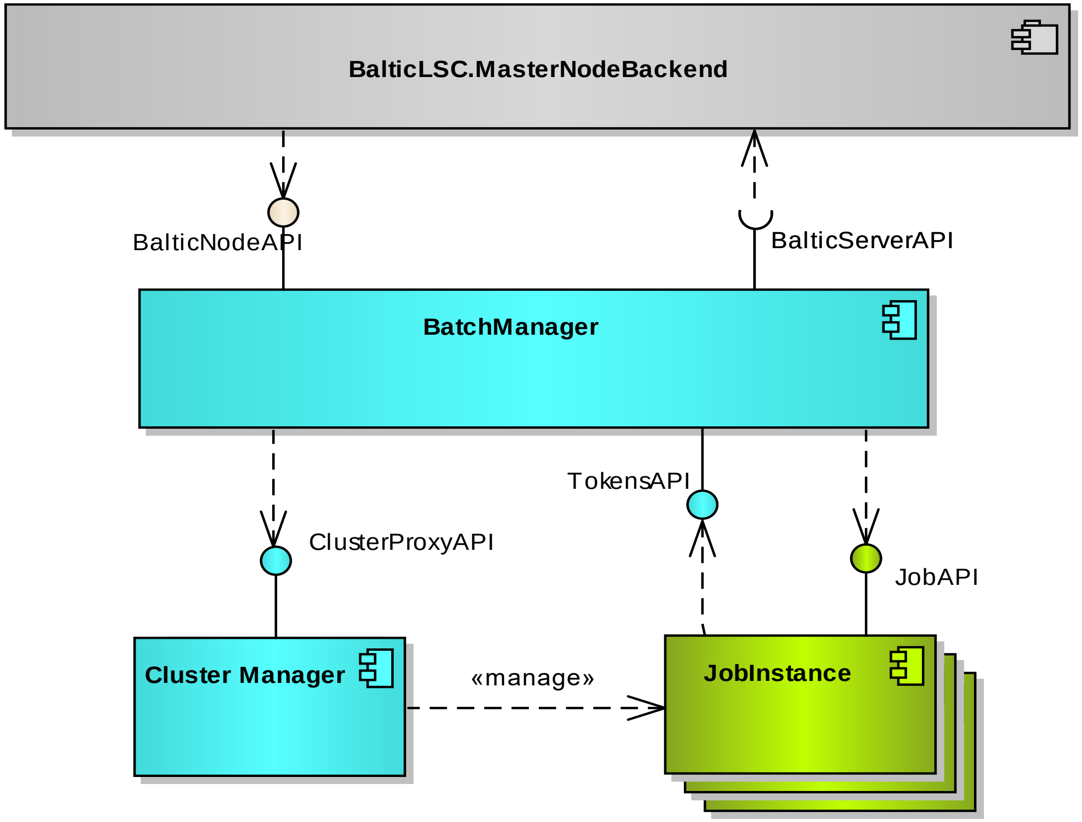


# How the BalticLSC Platform Works?



# :: Development of computation modules::

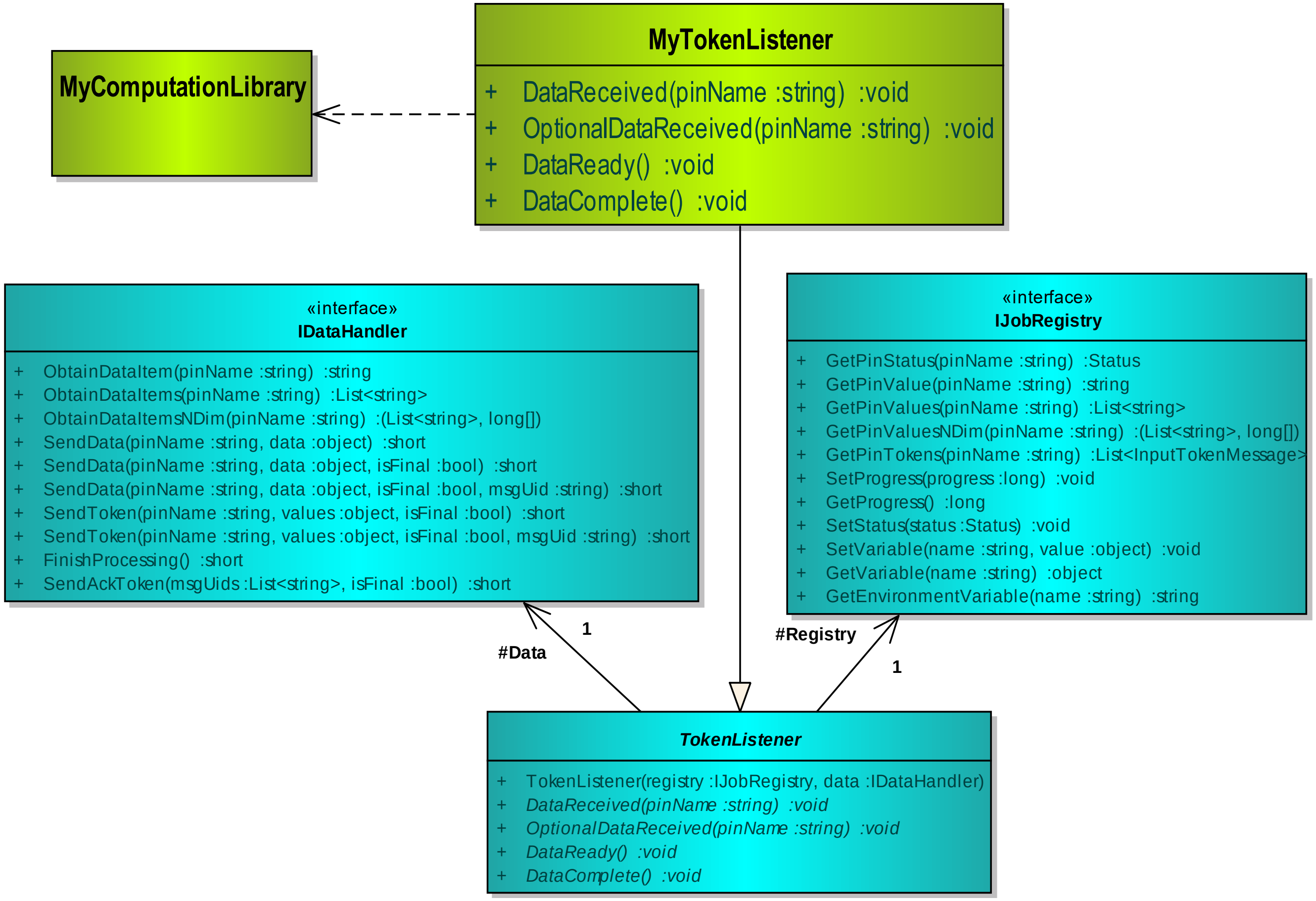
# Communication with comp. modules



- Computation module (design time) → Job Instance (runtime)
- Sending data tokens through JobAPI and TokensAPI

# Programming with a template (SDK)

- Develop “MyTokenListener”
- Use two standard interfaces “IDataHandle” and “IJobRegistry”



**MyComputationLibrary**

**MyTokenListener**

+ DataReceived(pinName :string) :void
+ OptionalDataReceived(pinName :string) :void
+ DataReady() :void
+ DataComplete() :void

«interface»
**IDataHandler**

+ ObtainDataItem(pinName :string) :string
+ ObtainDataItems(pinName :string) :List<string>
+ ObtainDataItemsNDim(pinName :string) :(List<string>, long[])
+ SendData(pinName :string, data :object) :short
+ SendData(pinName :string, data :object, isFinal :bool) :short
+ SendData(pinName :string, data :object, isFinal :bool, msgUid :string) :short
+ SendToken(pinName :string, values :object, isFinal :bool) :short
+ SendToken(pinName :string, values :object, isFinal :bool, msgUid :string) :short
+ FinishProcessing() :short
+ SendAckToken(msgUids :List<string>, isFinal :bool) :short

«interface»
**IJobRegistry**

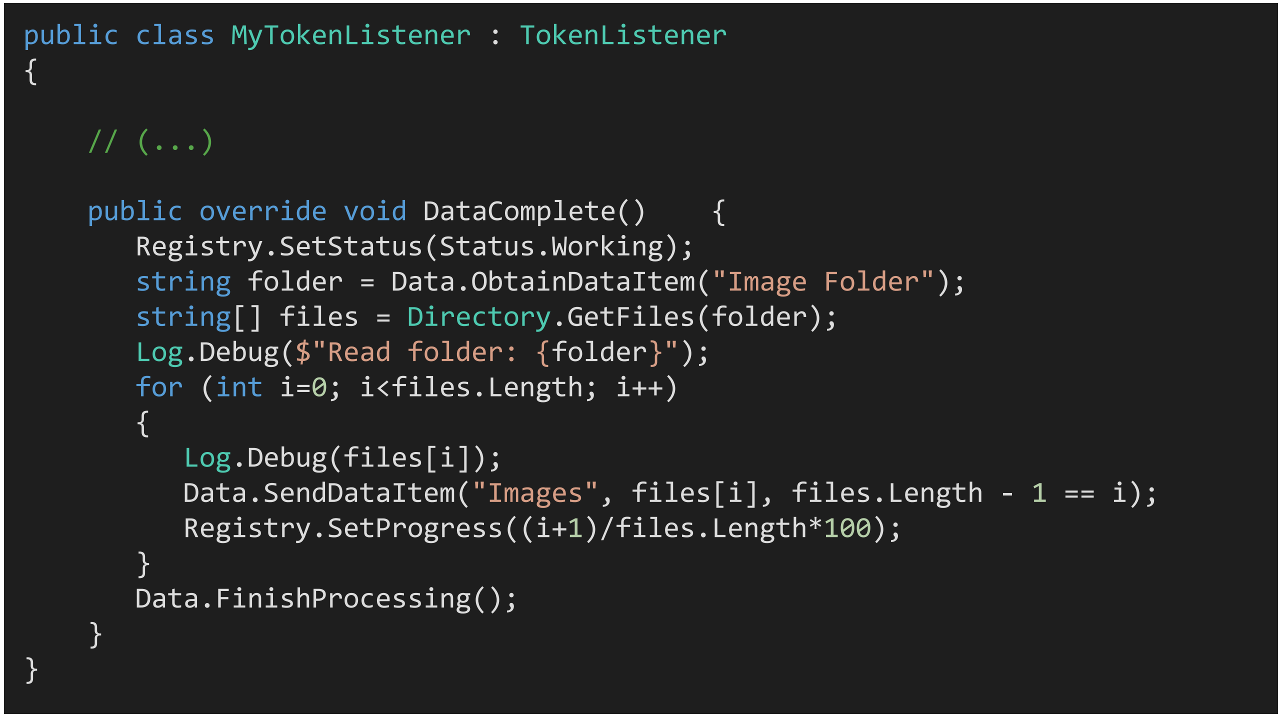
+ GetPinStatus(pinName :string) :Status
+ GetPinValue(pinName :string) :string
+ GetPinValues(pinName :string) :List<string>
+ GetPinValuesNDim(pinName :string) :(List<string>, long[])
+ GetPinTokens(pinName :string) :List<InputTokenMessage>
+ SetProgress(progress :long) :void
+ GetProgress() :long
+ SetStatus(status :Status) :void
+ SetVariable(name :string, value :object) :void
+ GetVariable(name :string) :object
+ GetEnvironmentVariable(name :string) :string

***TokenListener***

+ TokenListener(registry :IJobRegistry, data :IDataHandler)
+ *DataReceived(pinName :string) :void*
+ *OptionalDataReceived(pinName :string) :void*
+ *DataReady() :void*
+ *DataComplete() :void*

# Providing your computation code

- Read data items from tokens
- Perform computations
- Write data items and pass tokens



```
public class MyTokenListener : TokenListener
{

    // (...)

    public override void DataComplete()    {
       Registry.SetStatus(Status.Working);
       string folder = Data.ObtainDataItem("Image Folder");
       string[] files = Directory.GetFiles(folder);
       Log.Debug($"Read folder: {folder}");
       for (int i=0; i<files.Length; i++)
       {
          Log.Debug(files[i]);
          Data.SendDataItem("Images", files[i], files.Length - 1 == i);
          Registry.SetProgress((i+1)/files.Length*100);
       }
       Data.FinishProcessing();
    }
}
```

# BalticLSC Developer’s Manual

BalticLSC Developer's Manual

How to develop BalticLSC Applications and Modules – detailed instructions for software developers

Version 0.14

Priority 1: Innovation

- Programming in CAL
- Programming computation modules
- Download from:

https://github.com/balticlsc/Documents

Warsaw University
of Technology

BalticLSC

## Contact

BalticLSC Secretariat
balticlsc@ee.pw.edu.pl
tel. +48 22 234 7350
www.balticlsc.eu

Interreg
Baltic Sea Region

EUROPEAN UNION

EUROPEAN
REGIONAL
DEVELOPMENT
FUND