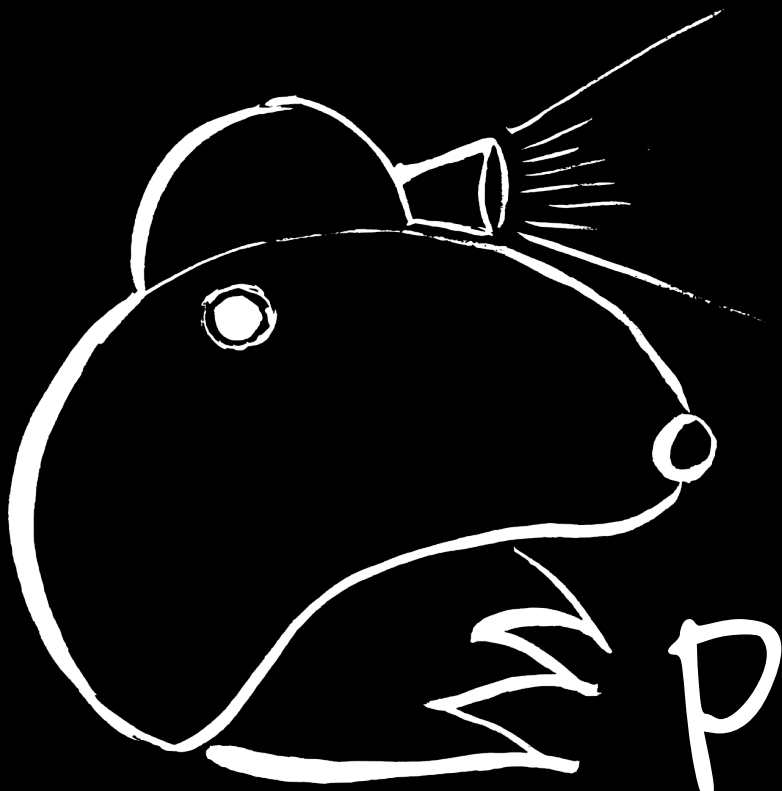
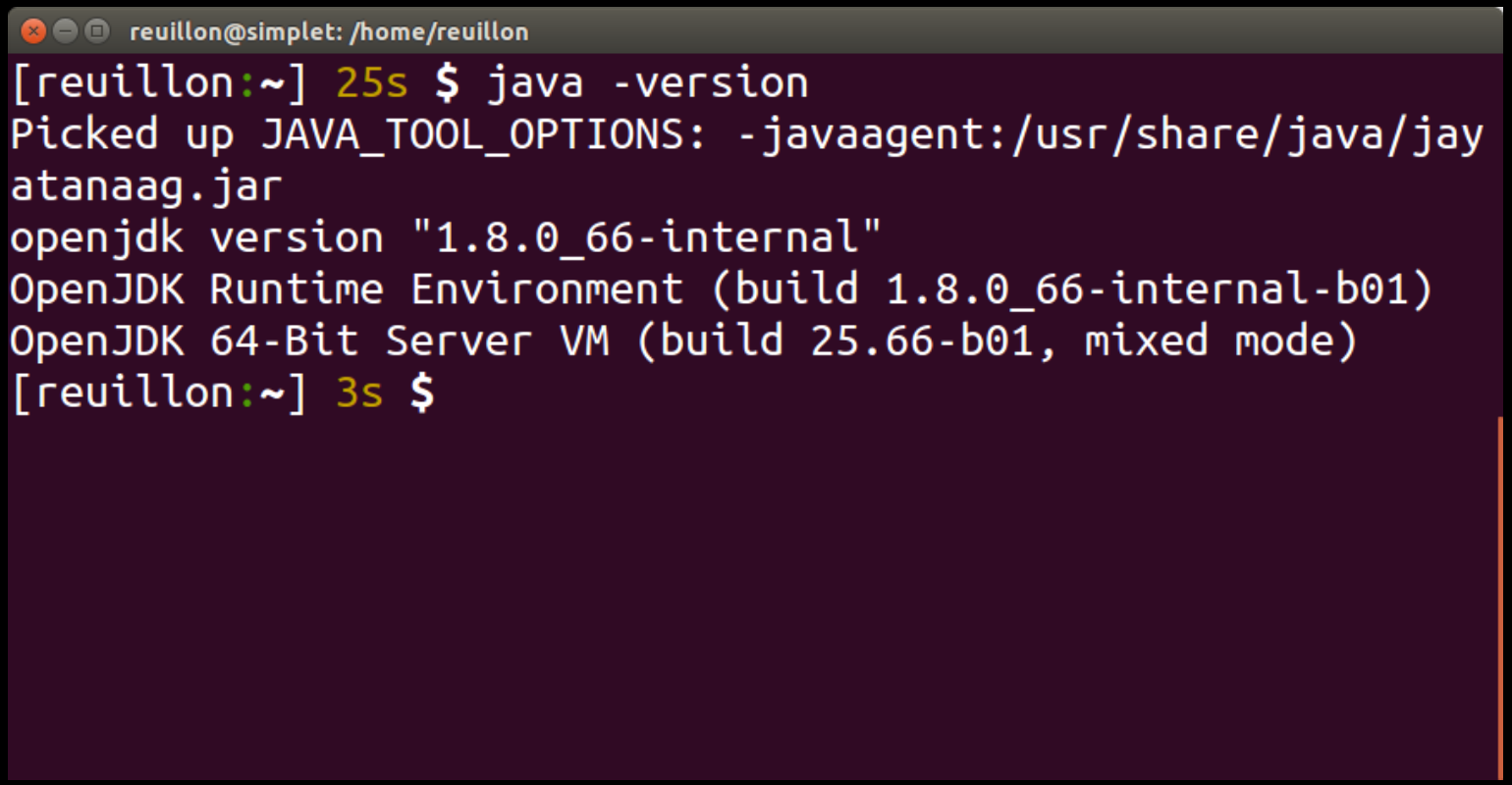
First steps with



OpenMOLE

Prerequisite

Make sure you have the Java runtime environment 7 or 8



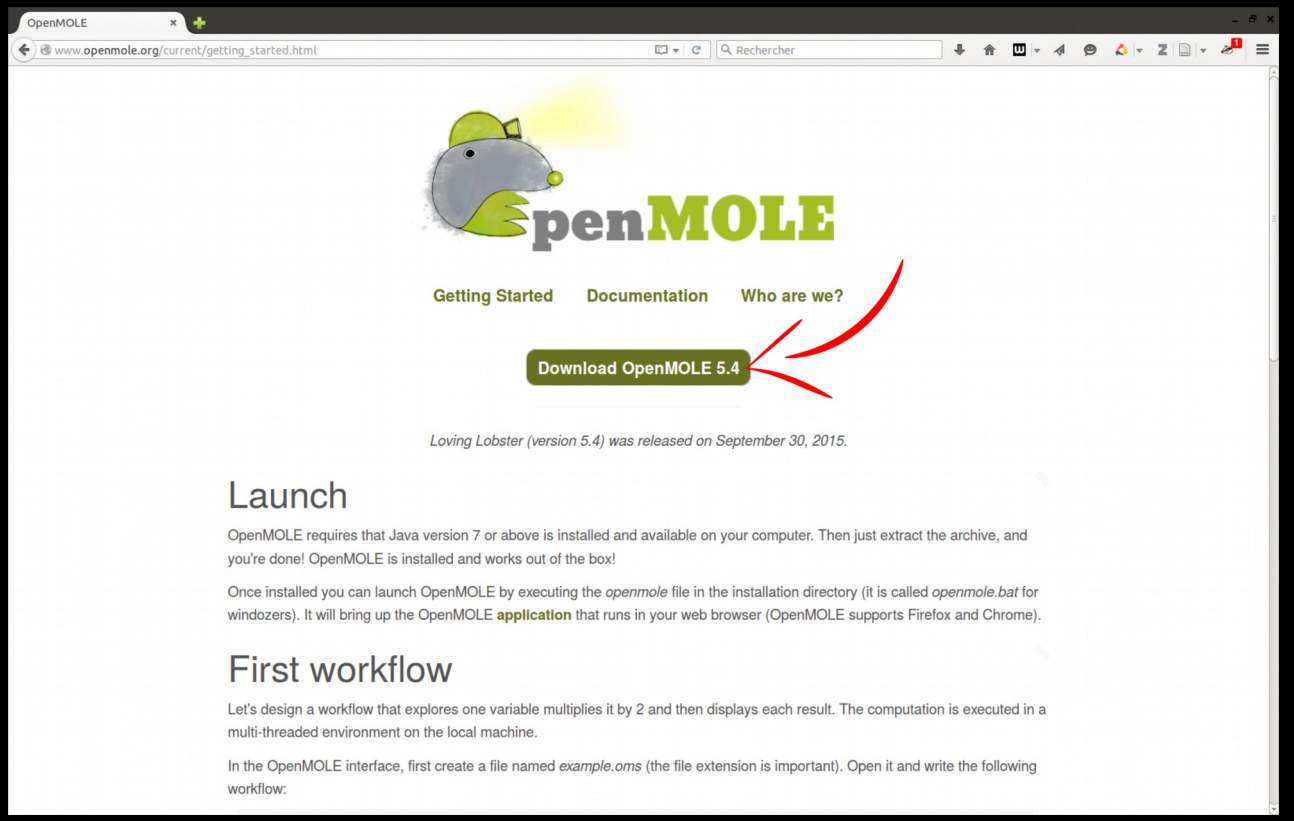
Prerequisite

Make sure you have Chrom(e | ium) or Firefox

Download

Go to: www.openmole.org

Go to: Getting Started.

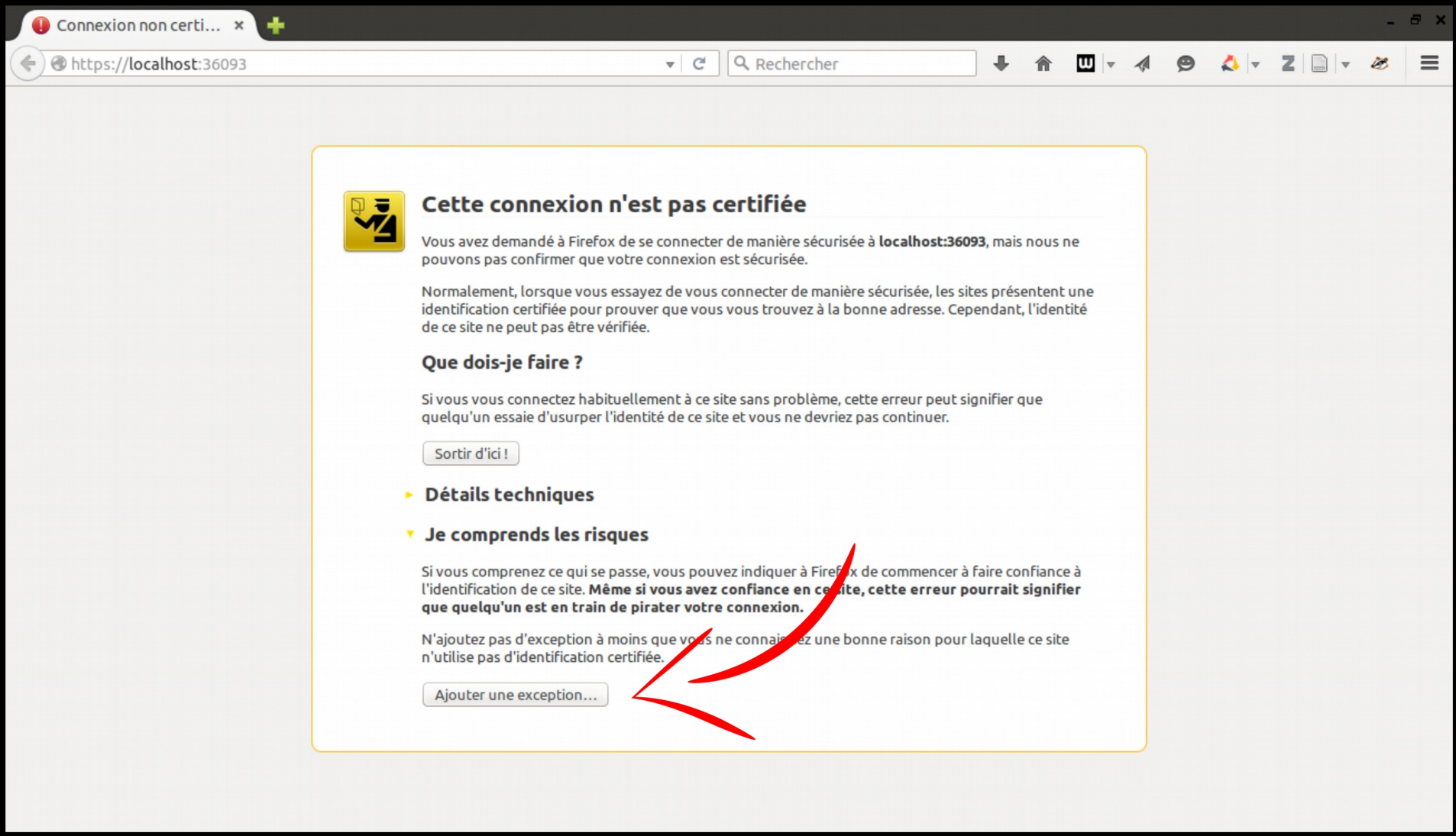


Launch

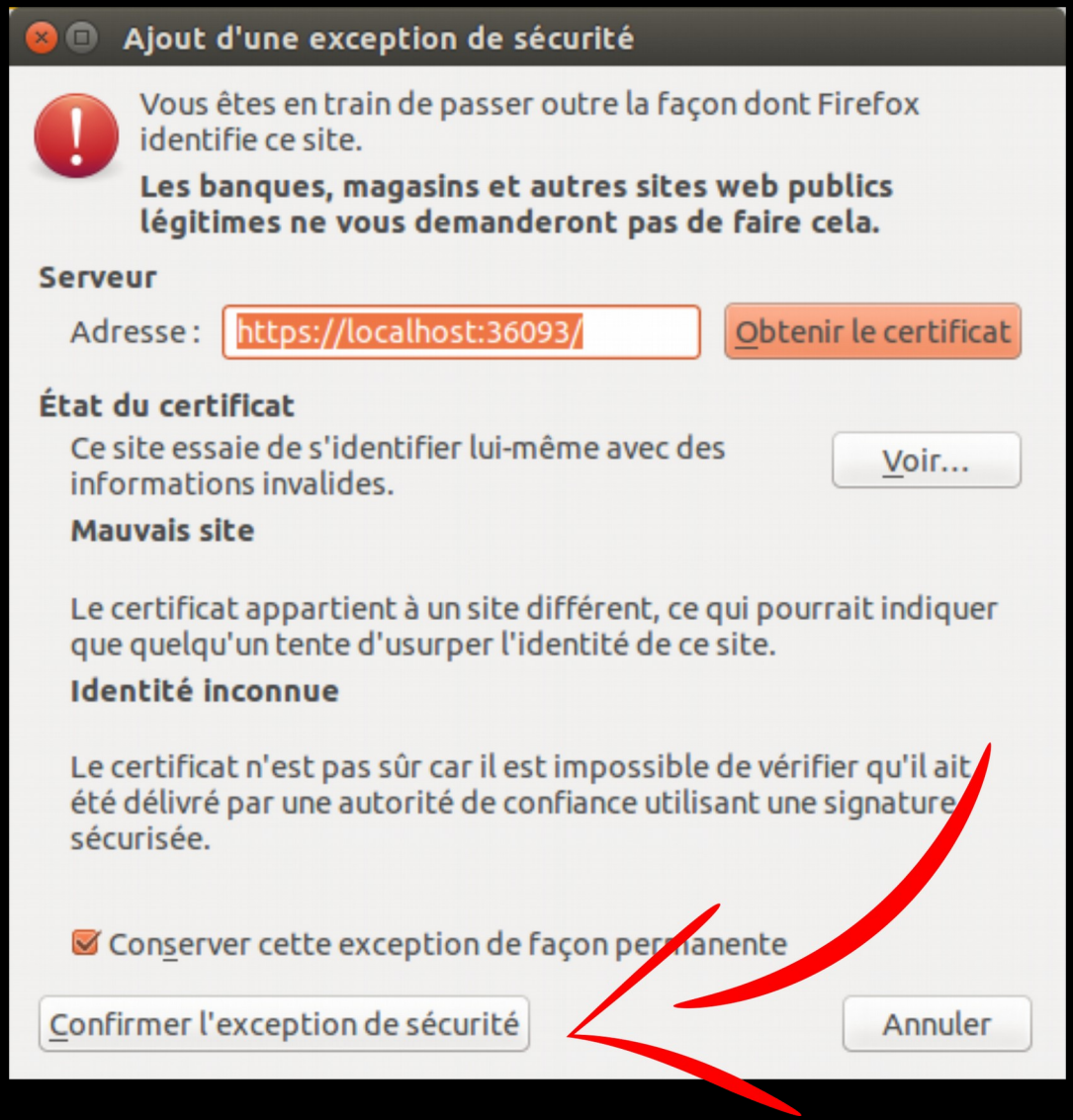
Run:

- Linux | Mac => openmole.sh
- Windoze => openmole.bat

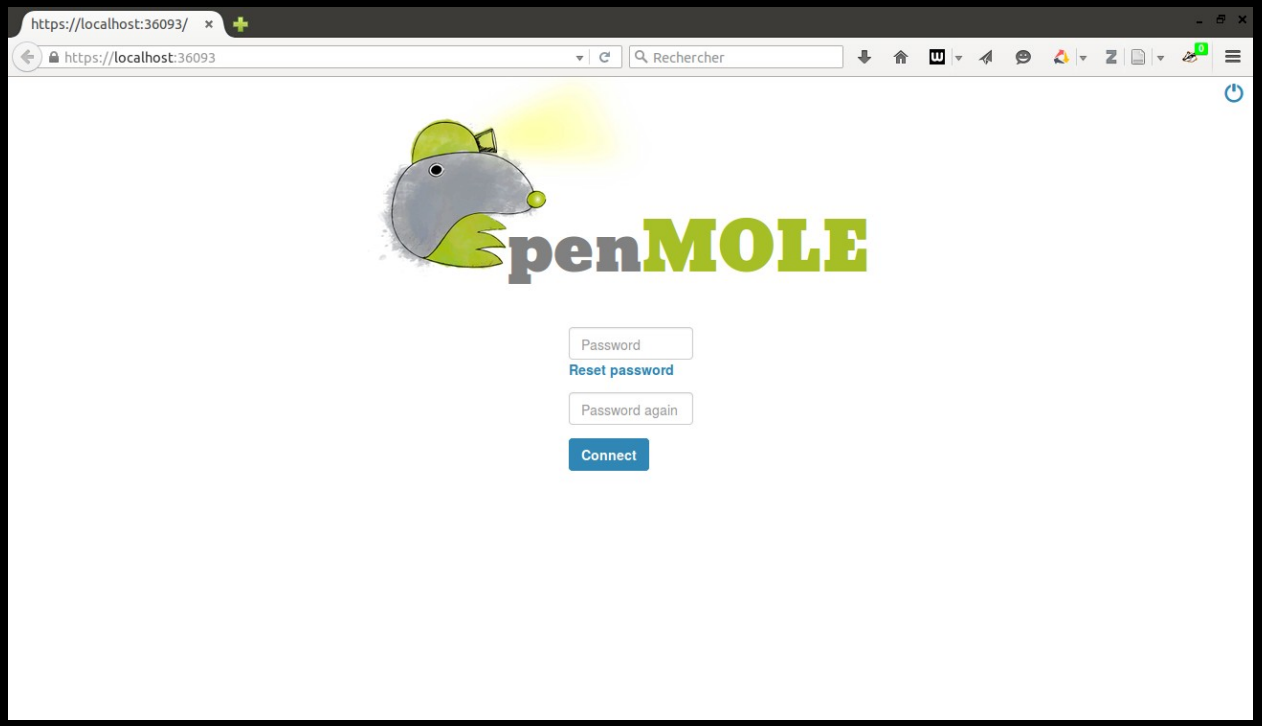
Insecure connection



Insecure connection

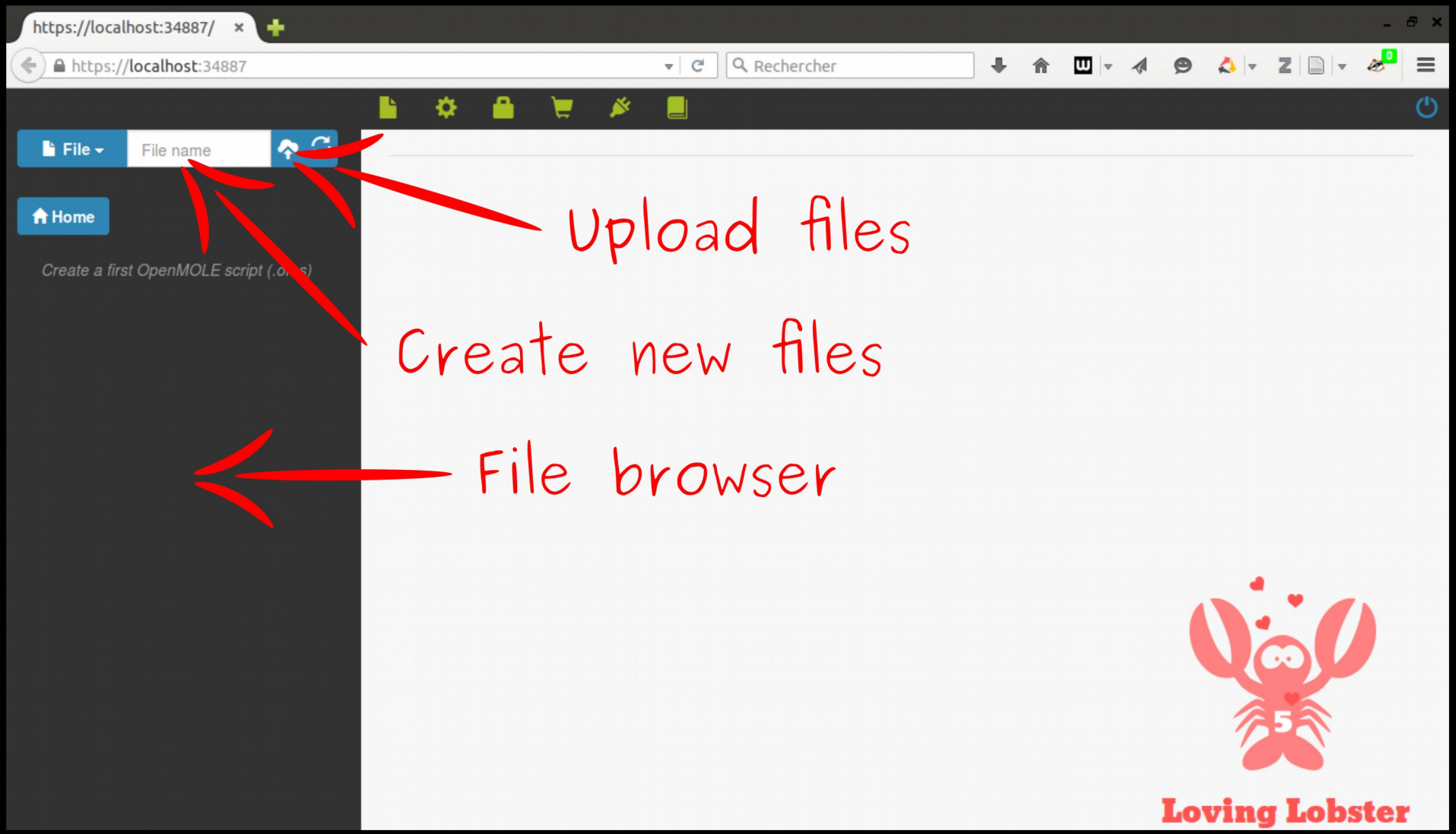


OpenMOLE password



Chose a password to encrypt secret data you provide to OpenMOLE: passwords for servers...

Welcome to OpenMOLE

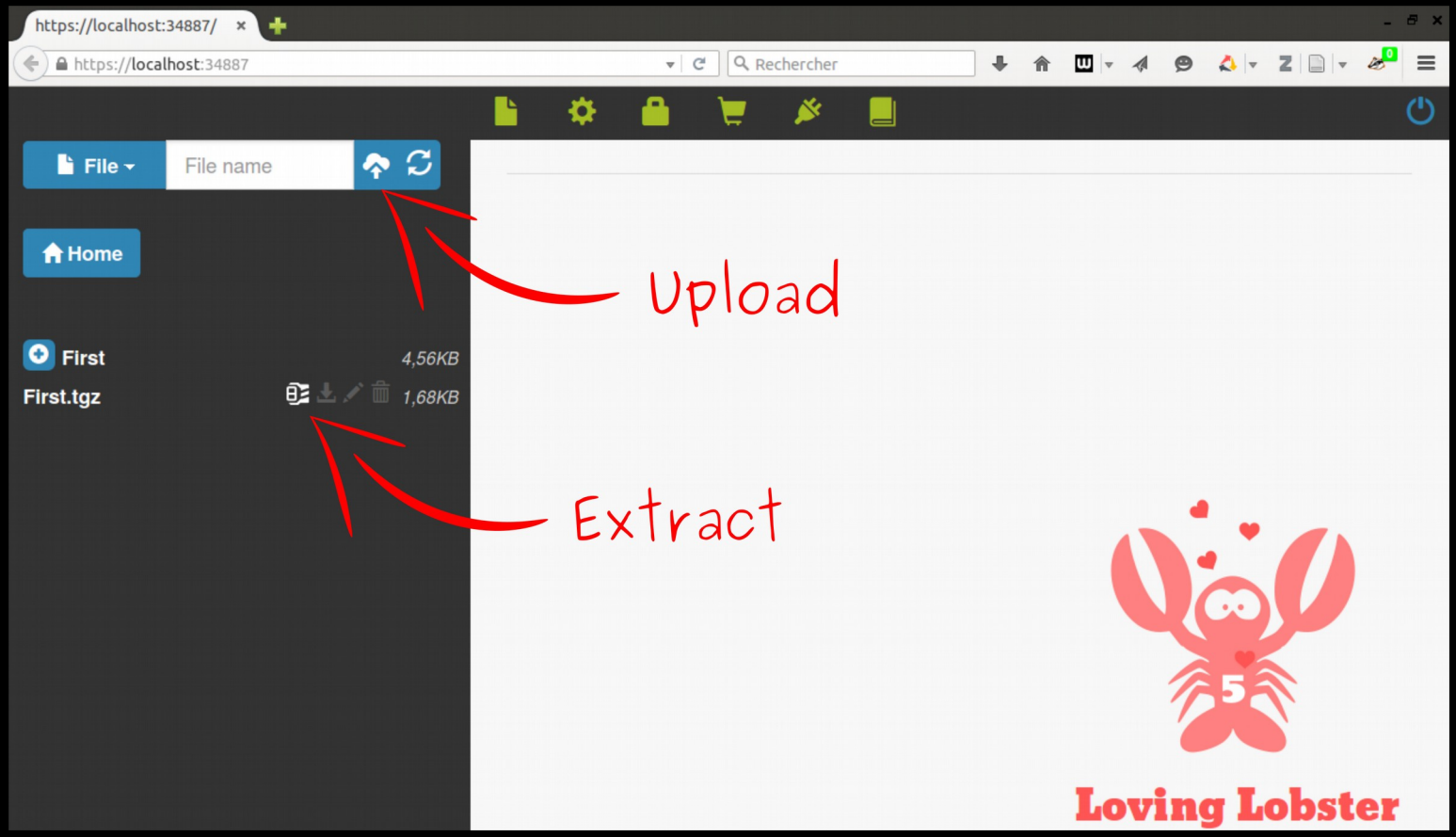


First workflow

Go to: www.openmole.org/files/csdc

Download First.tgz

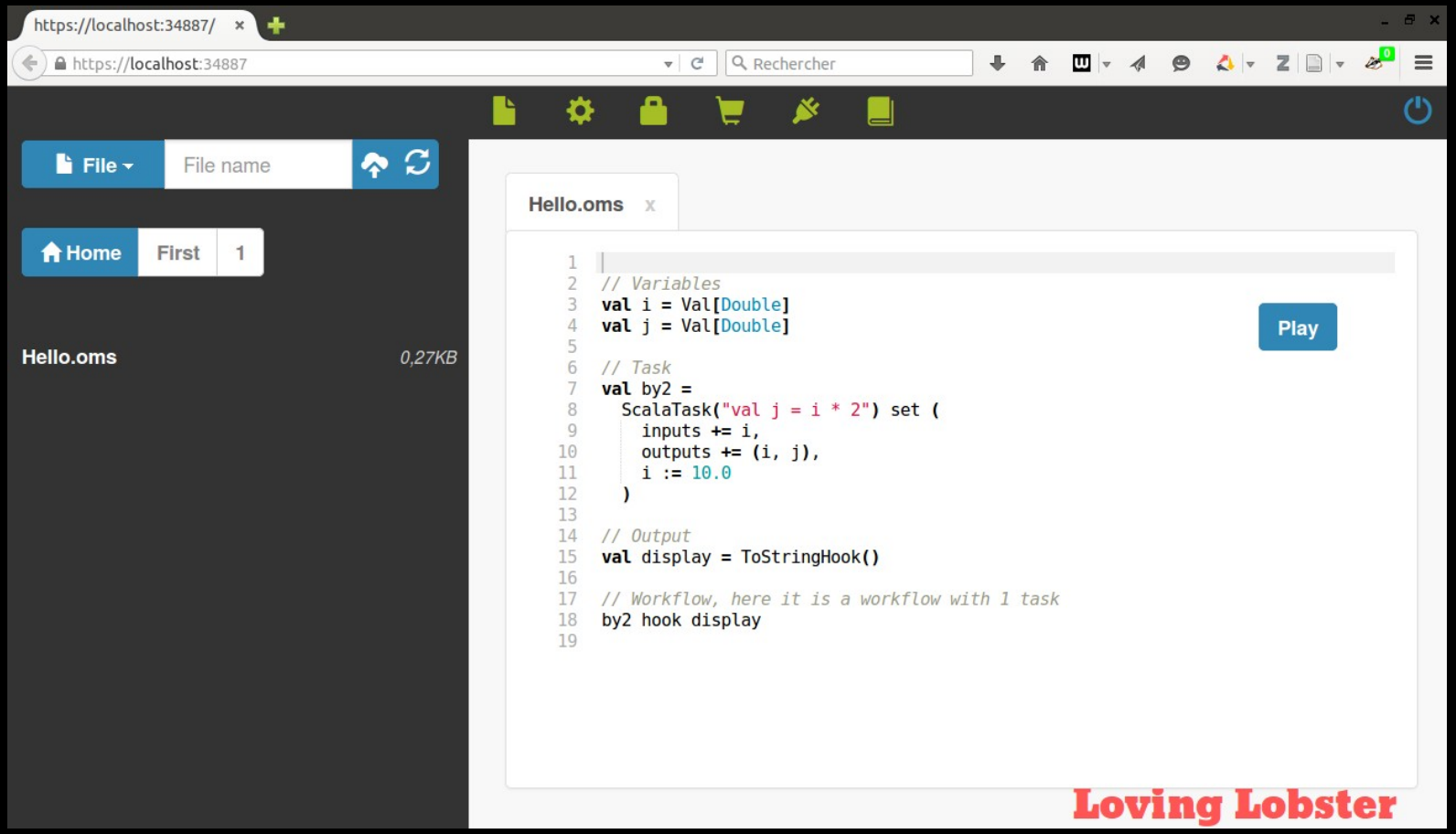
First workflow



Upload the archive

Extract it

First workflow

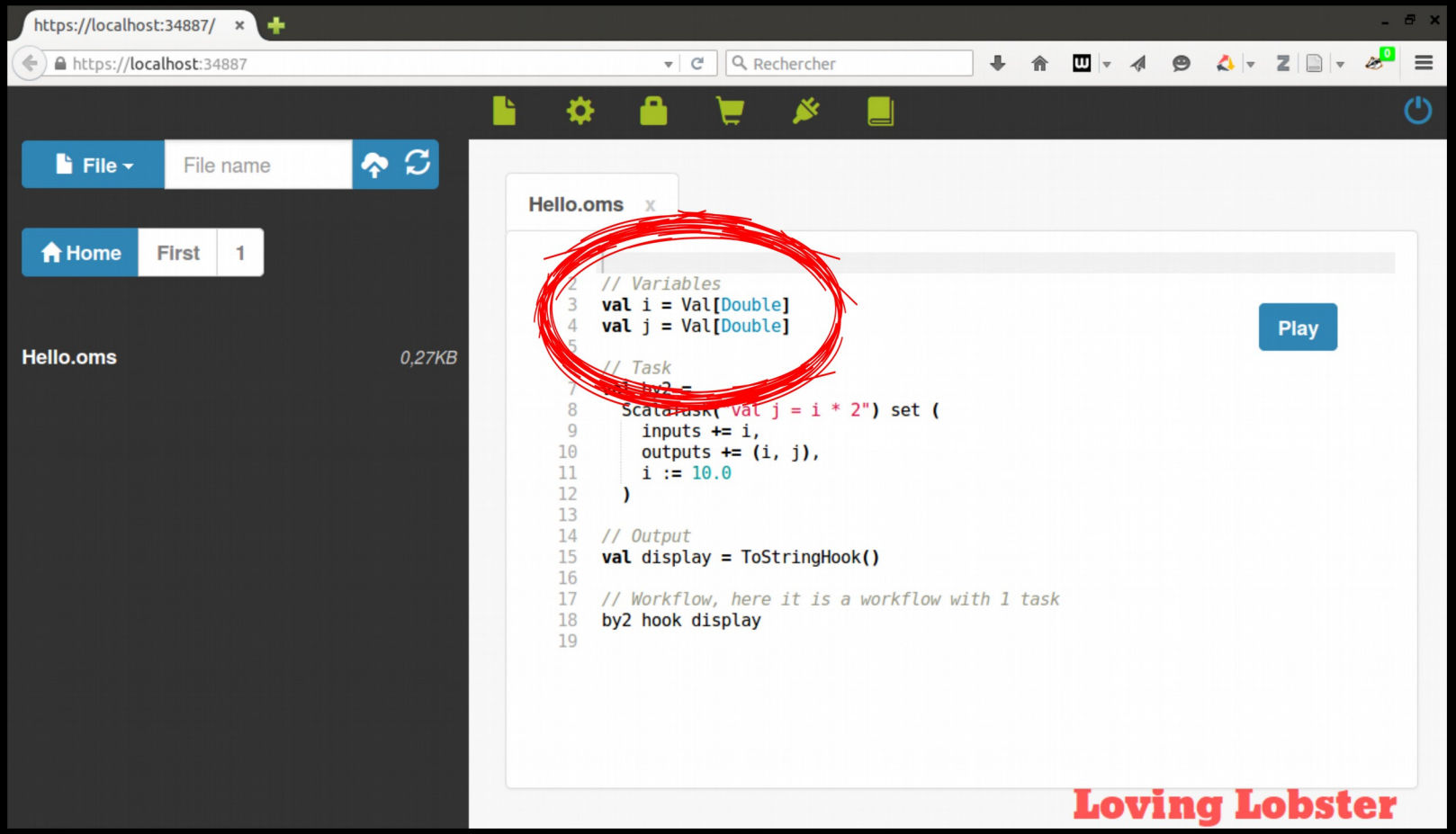


Click on the folder "First"

Click on the folder "1"

Click on the file "Hello.oms"

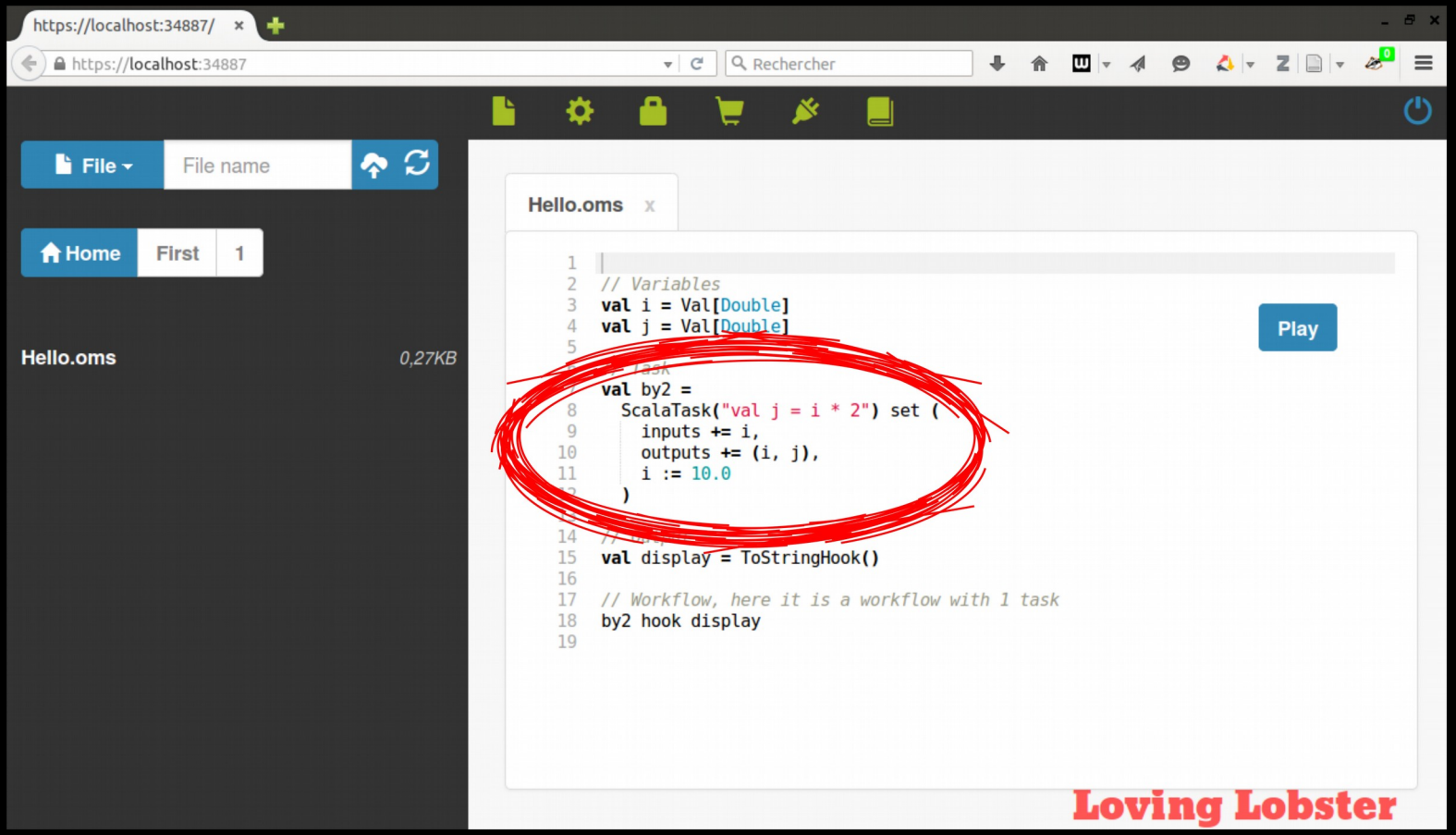
First workflow



Express the dataflow

Variables are typed

First workflow

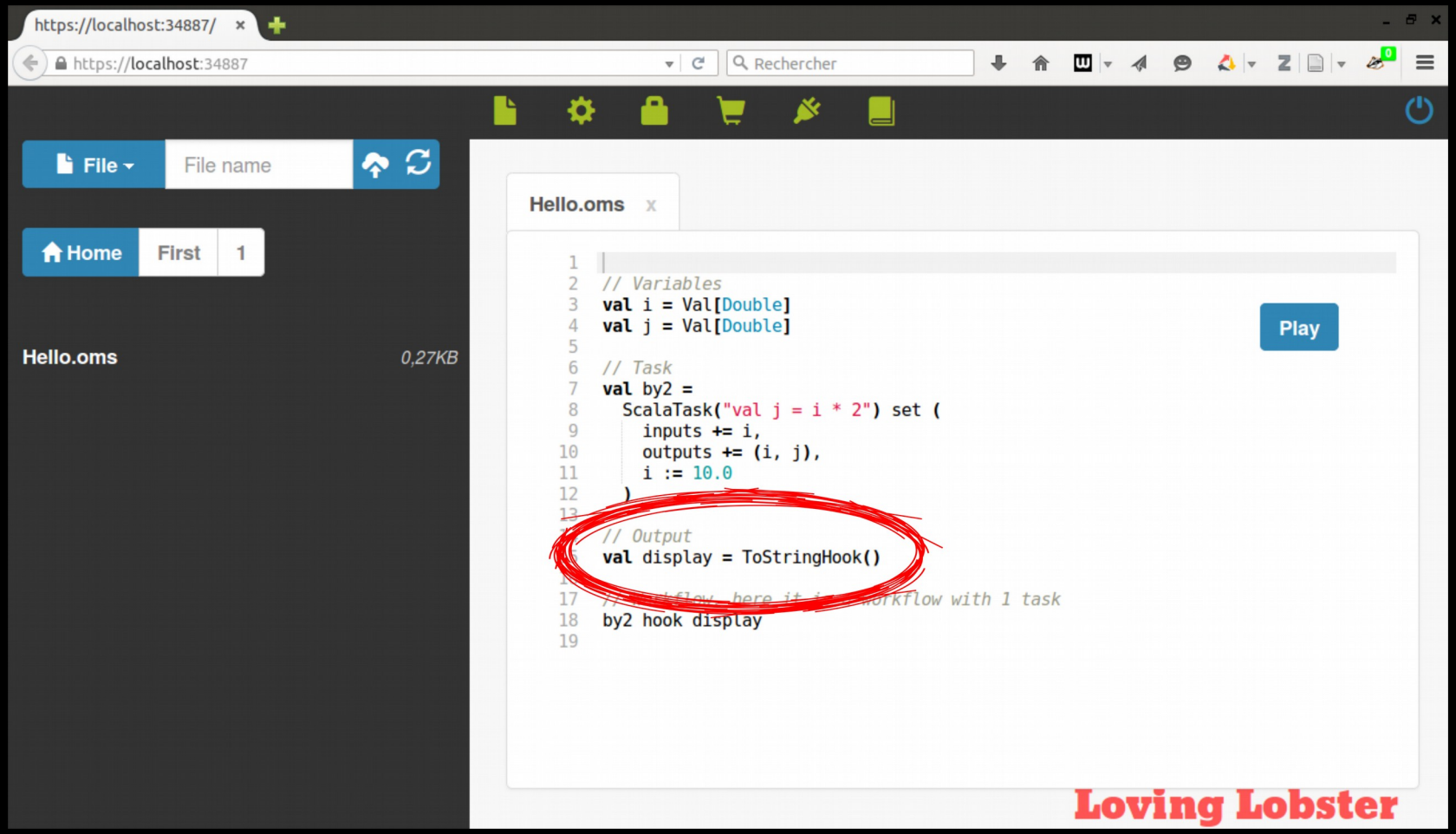


Express computation

Uses inputs, produces outputs

(may specify default values)

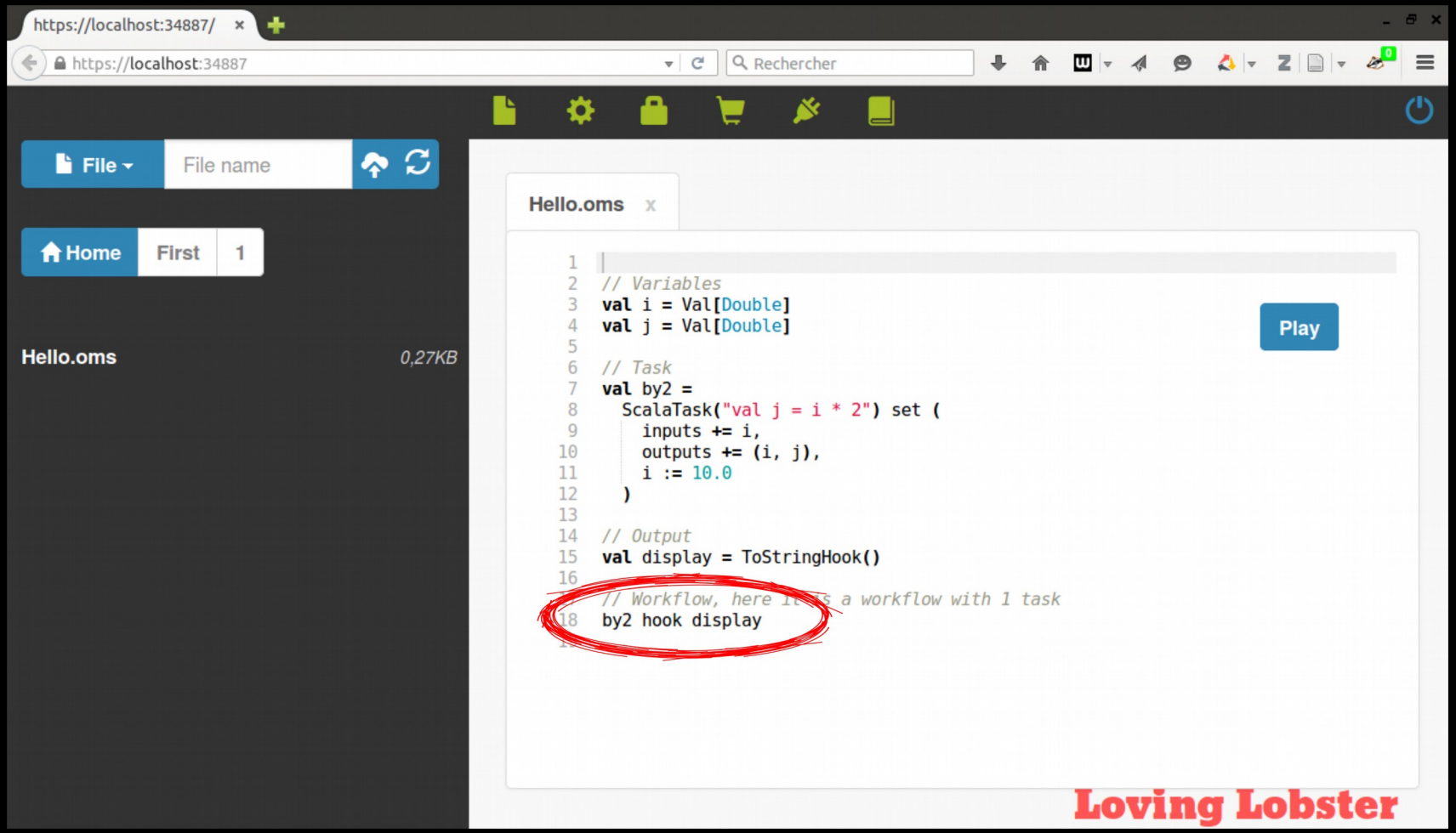
First workflow



Express output operations

Alway runs locally

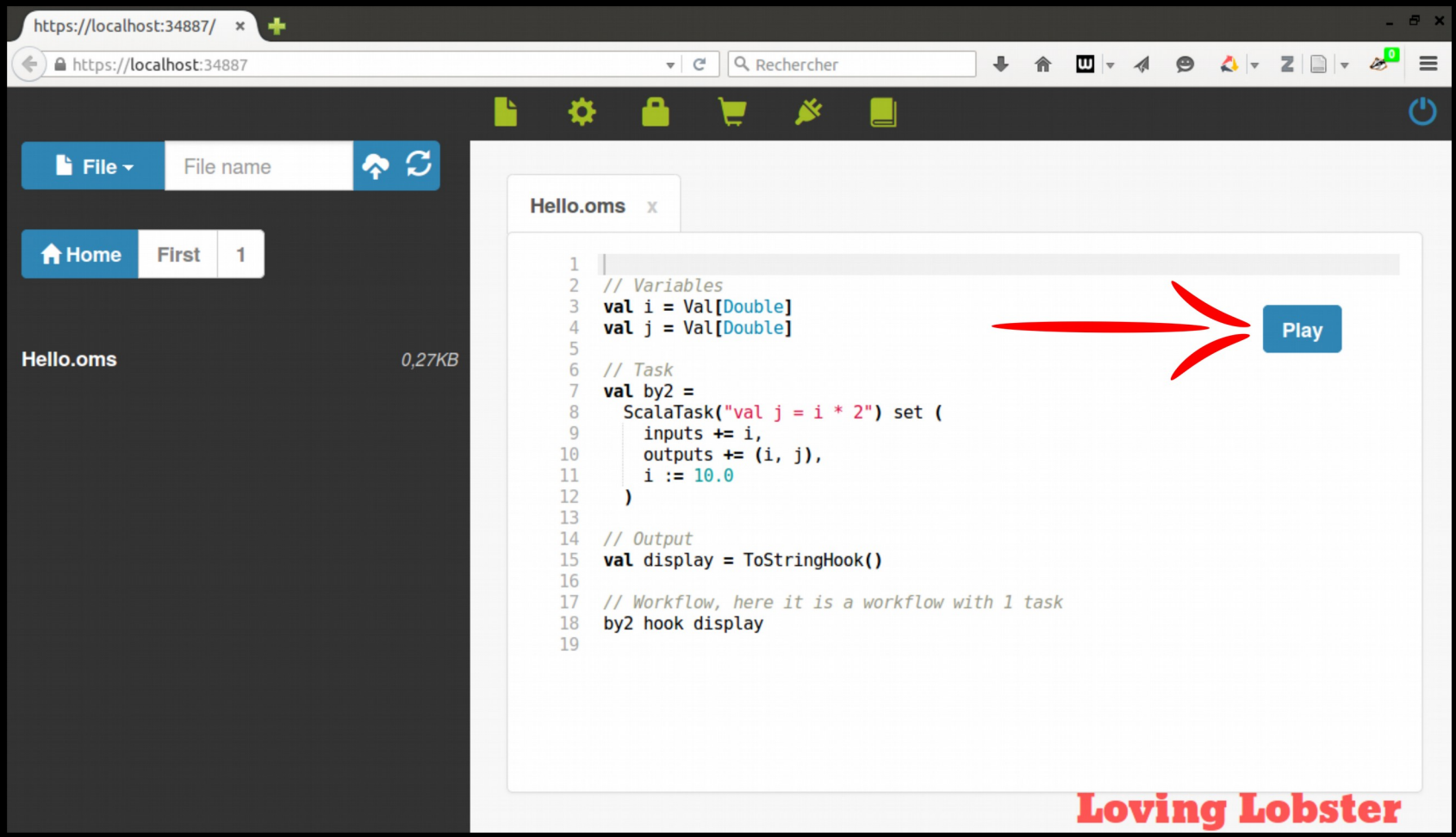
First workflow



Express workflow composition

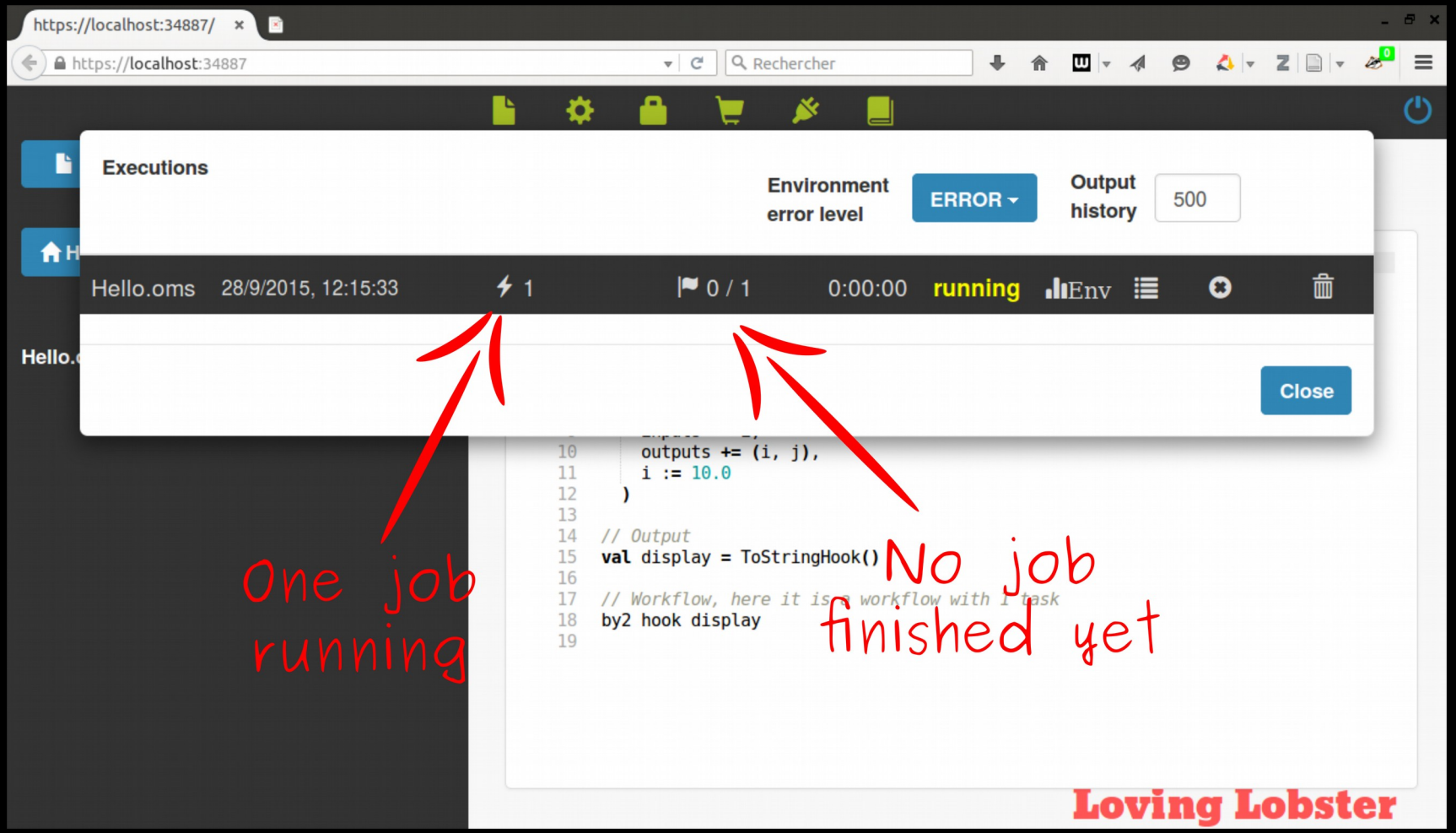
Here: One task + One hook

First workflow



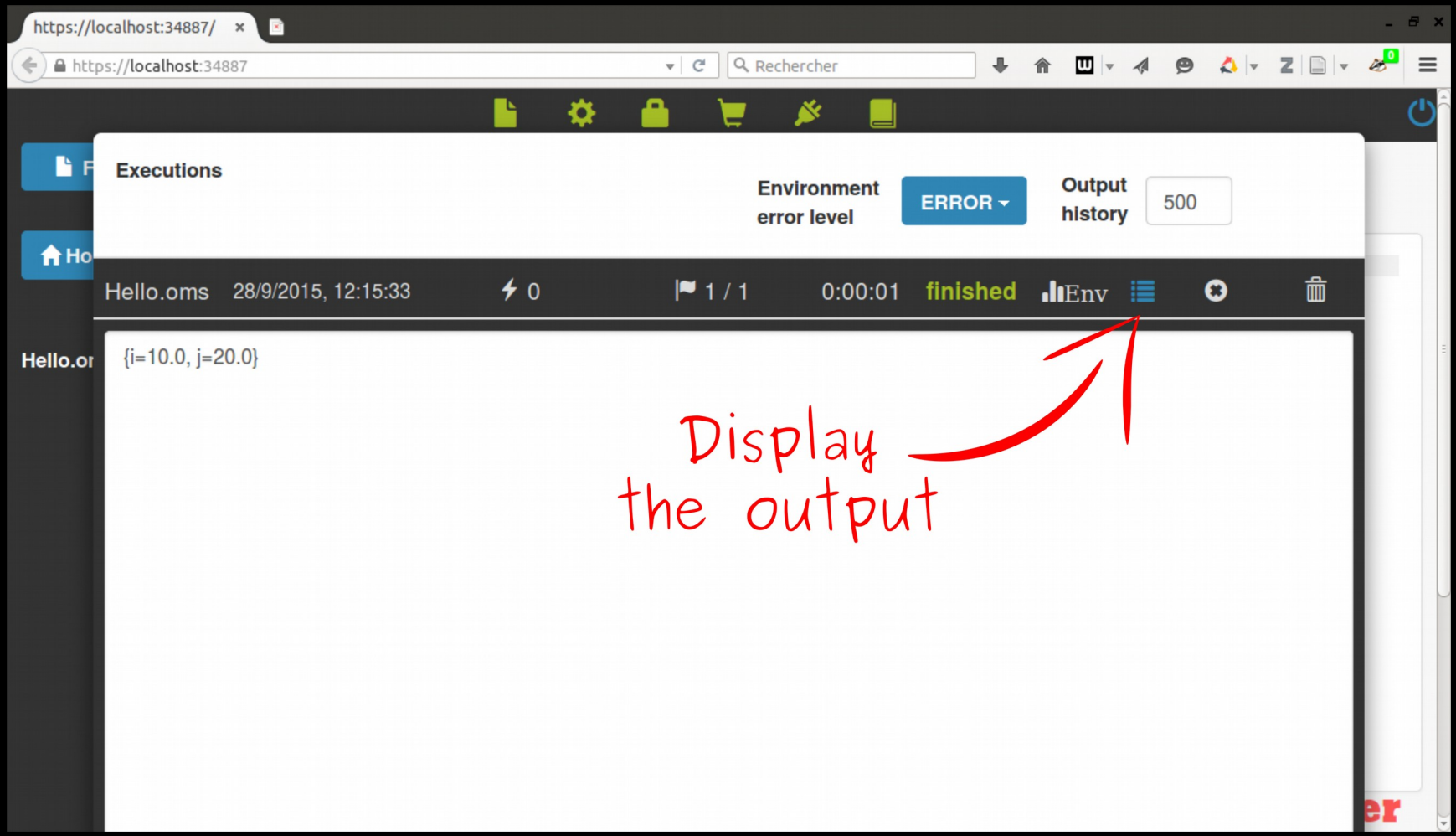
And now lets run!

First workflow



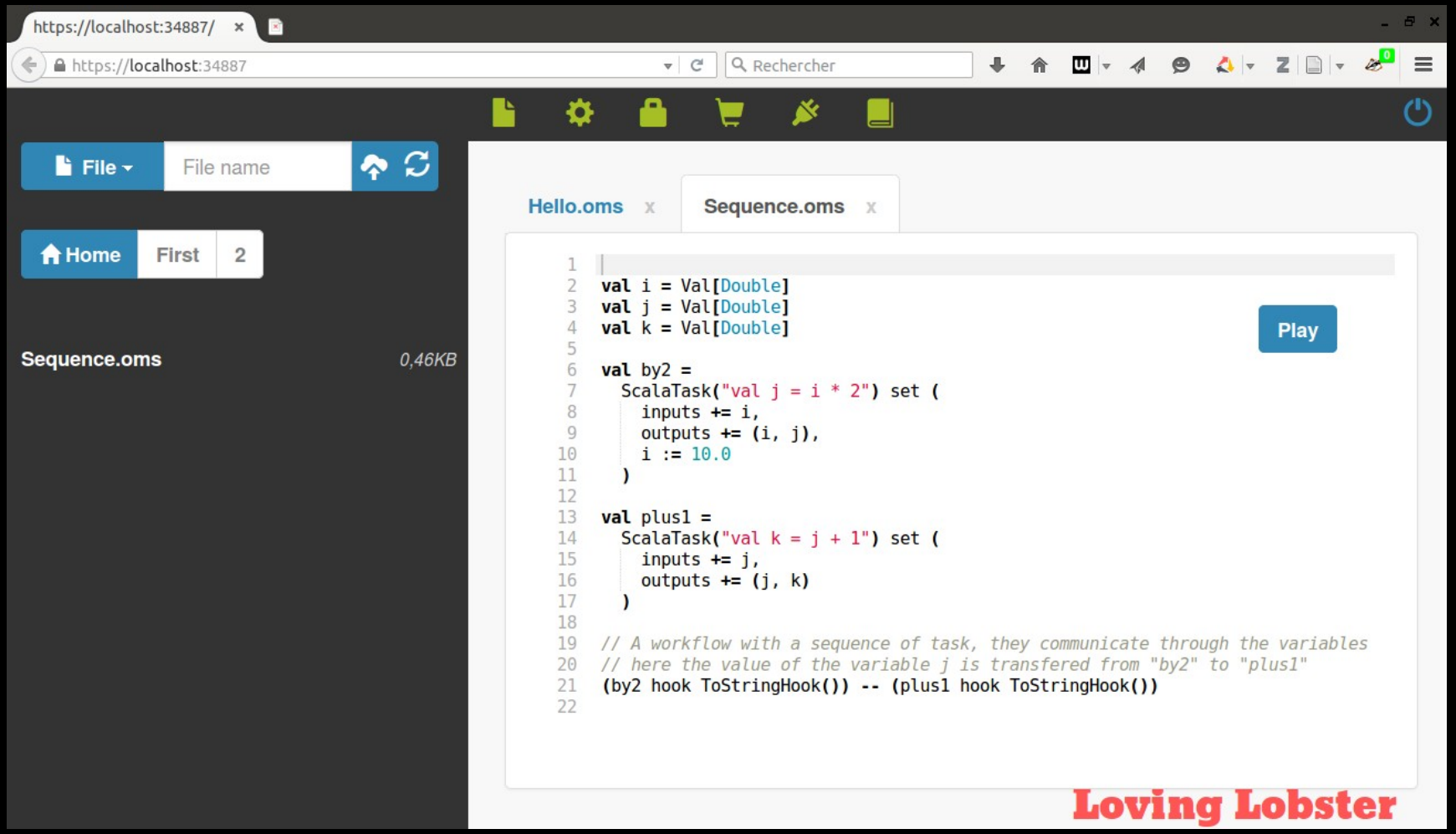
Running...

First workflow



Display the result.

The sequence

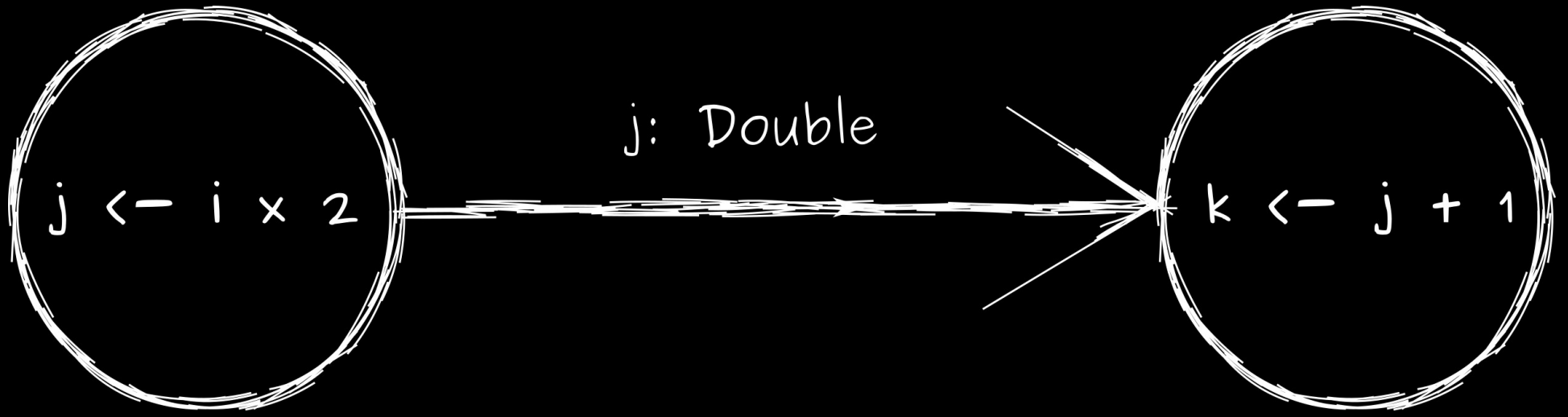


Go to the directory "2"

Click on "Sequence.oms"

The sequence

Tasks communicate through variables.

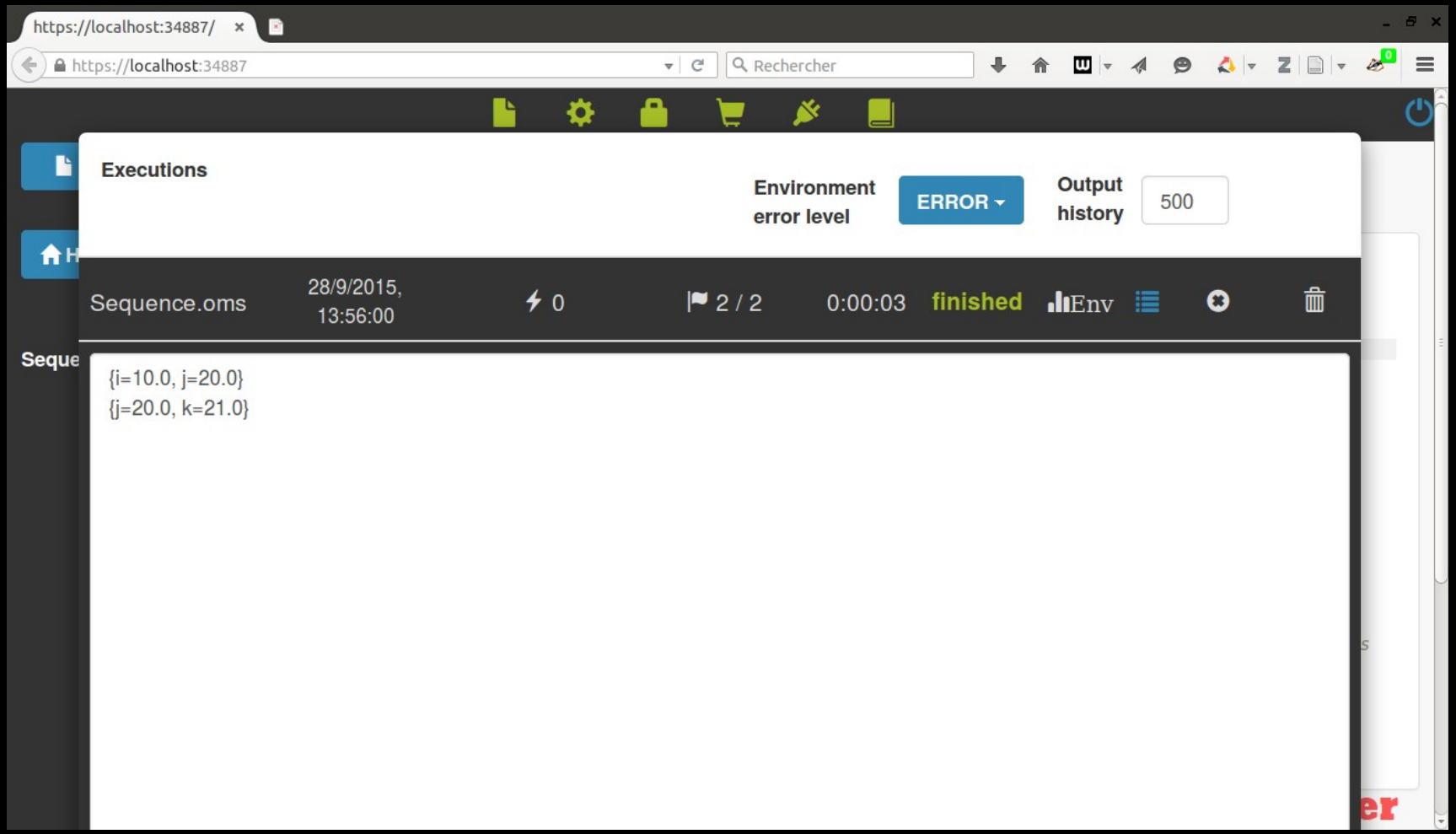


The sequence

A sequence of task is represented by --

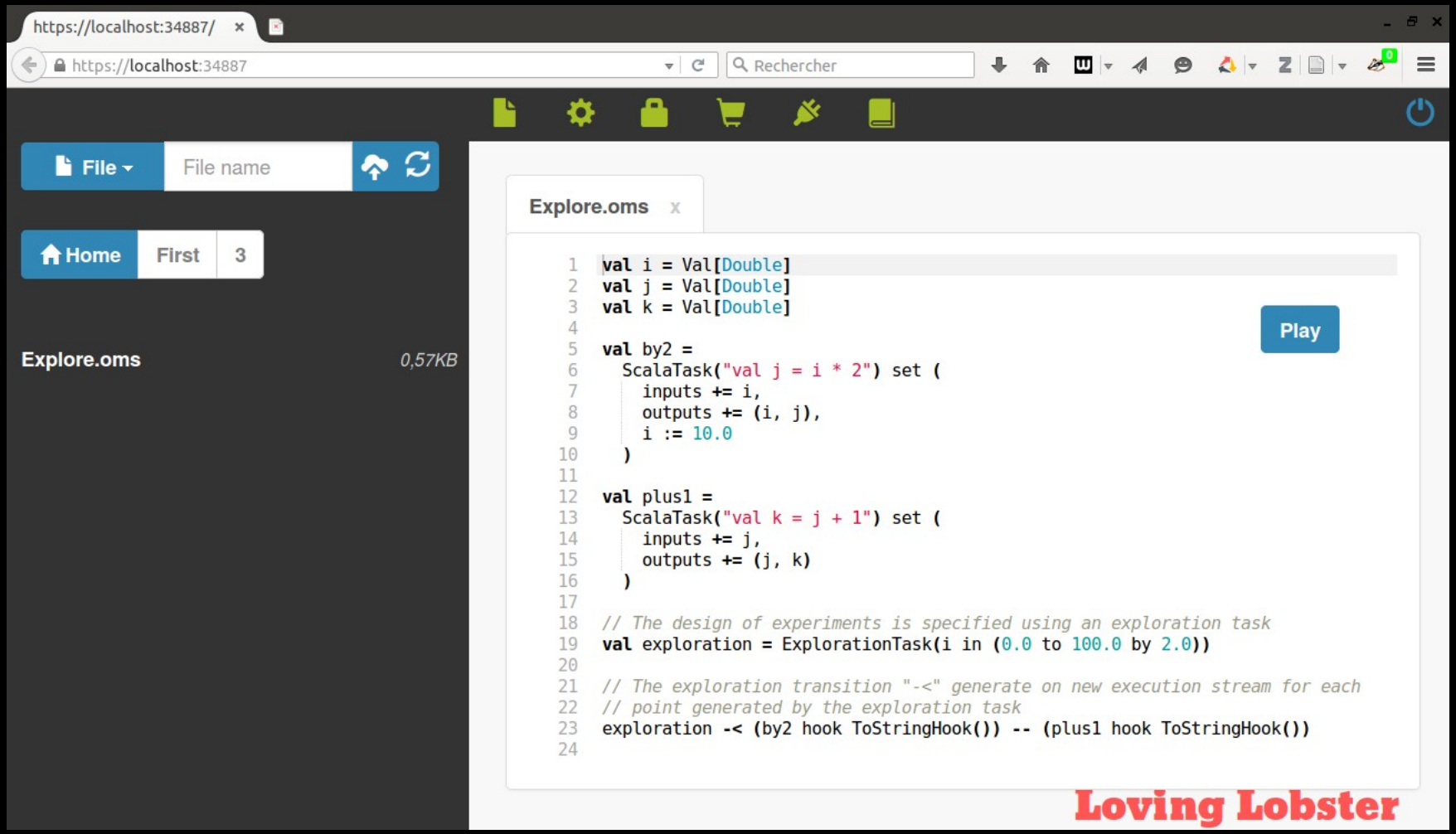
```
18
19  // A workflow with a sequence of task, they communicate through the variables
20  // here the value of the variable j is transfered from "by2" to "plus1"
21  (by2 hook ToStringHook()) -- (plus1 hook ToStringHook())
22
```

The sequence



Run the workflow.

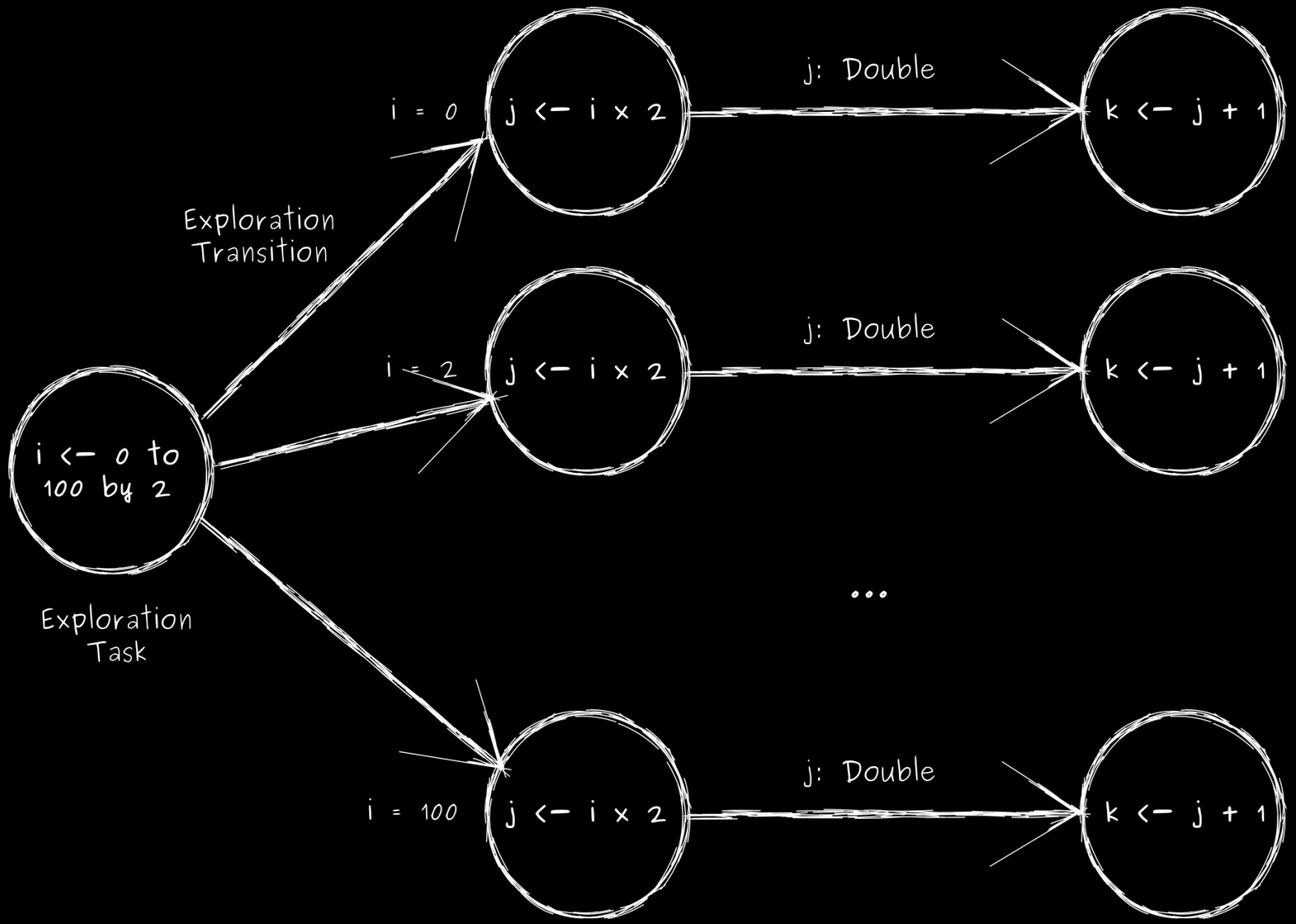
The exploration



Go to the directory "3"

Click on "Sequence.oms"

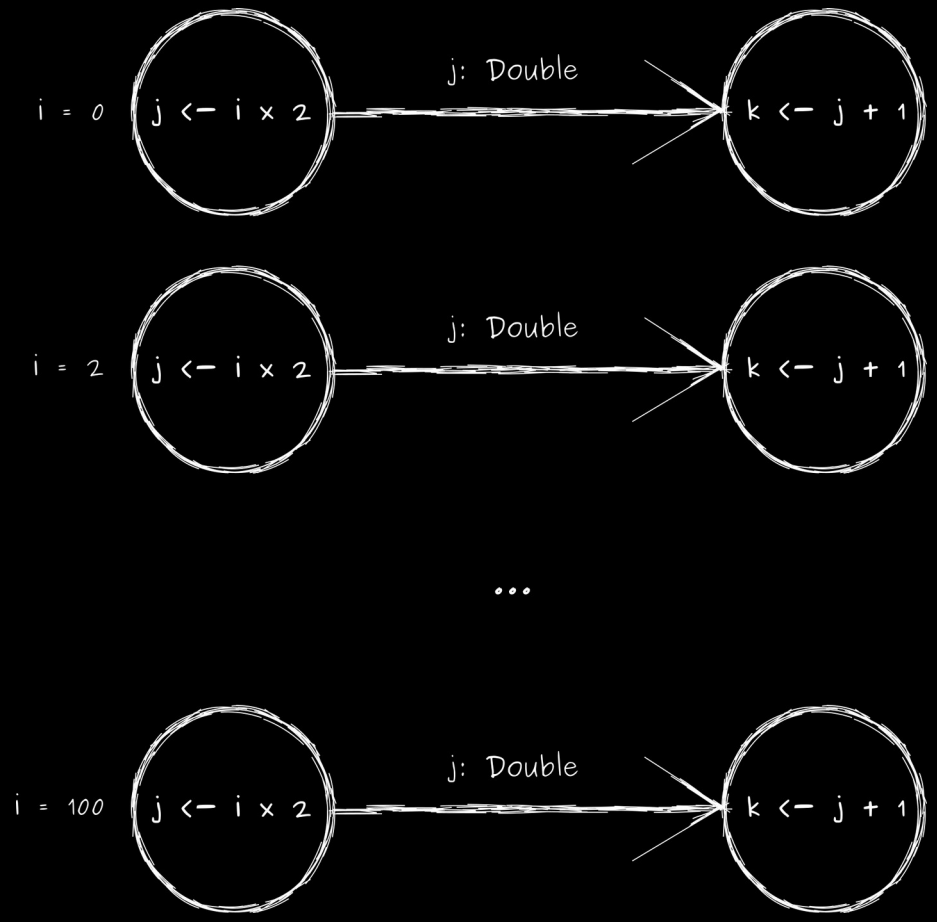
The exploration



Exploration Task + Exploration Transition

The exploration

The objective is to execute the sequence of tasks for many input values:



The exploration

```
18  // The design of experiments is specified using an exploration task
19  val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))
```

The exploration task contains the design of experiments.

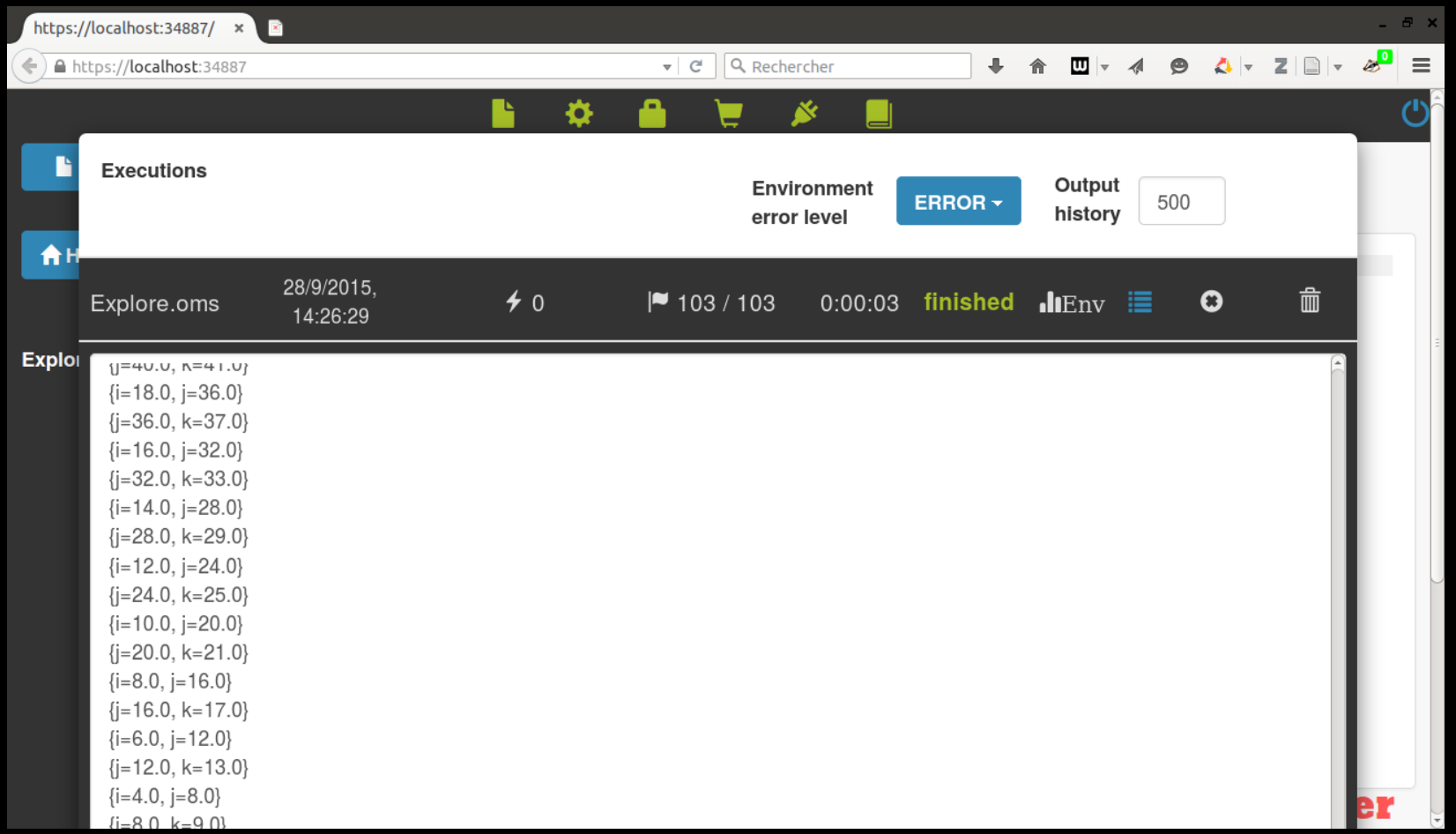
A design of experiments can compose many things (see the Sampling doc).

The exploration

```
21  // The exploration transition "-<" generate on new execution stream for each
22  // point generated by the exploration task
23  exploration -< (by2 hook ToStringHook()) -- (plus1 hook ToStringHook())
```

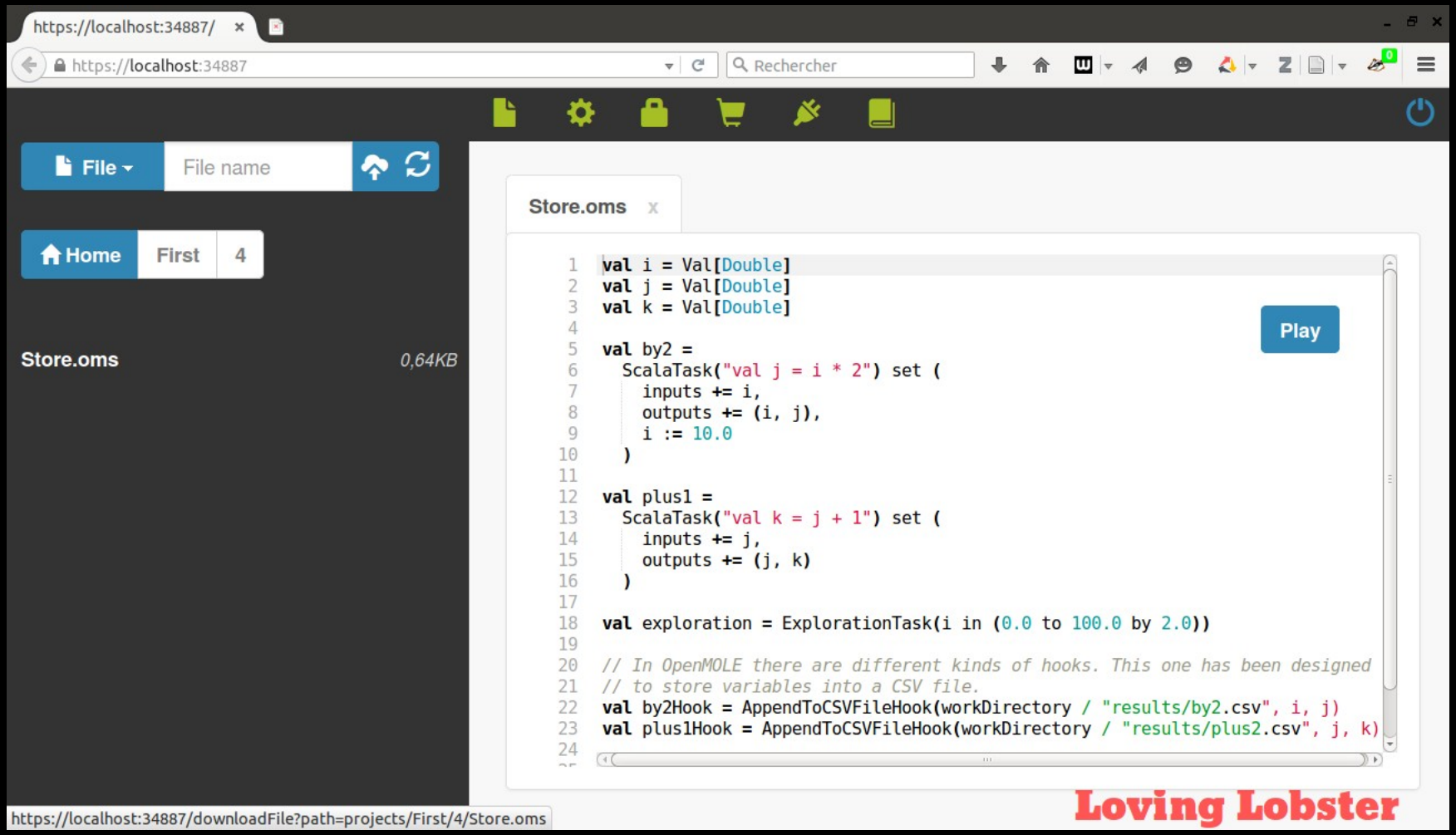
The notation for the exploration transition is "-<".

The exploration



Run it

Store results in files



Go to the directory "4"

Click on "Store.oms"

Store results in files

```
20  // In OpenMOLE there are different kinds of hooks. This one has been designed
21  // to store variables into a CSV file.
22  val by2Hook = AppendToCSVFileHook(workDirectory / "results/by2.csv", i, j)
23  val plus1Hook = AppendToCSVFileHook(workDirectory / "results/plus2.csv", j, k)
```

The "AppendToCSVFileHook" stores results in CSV Files.

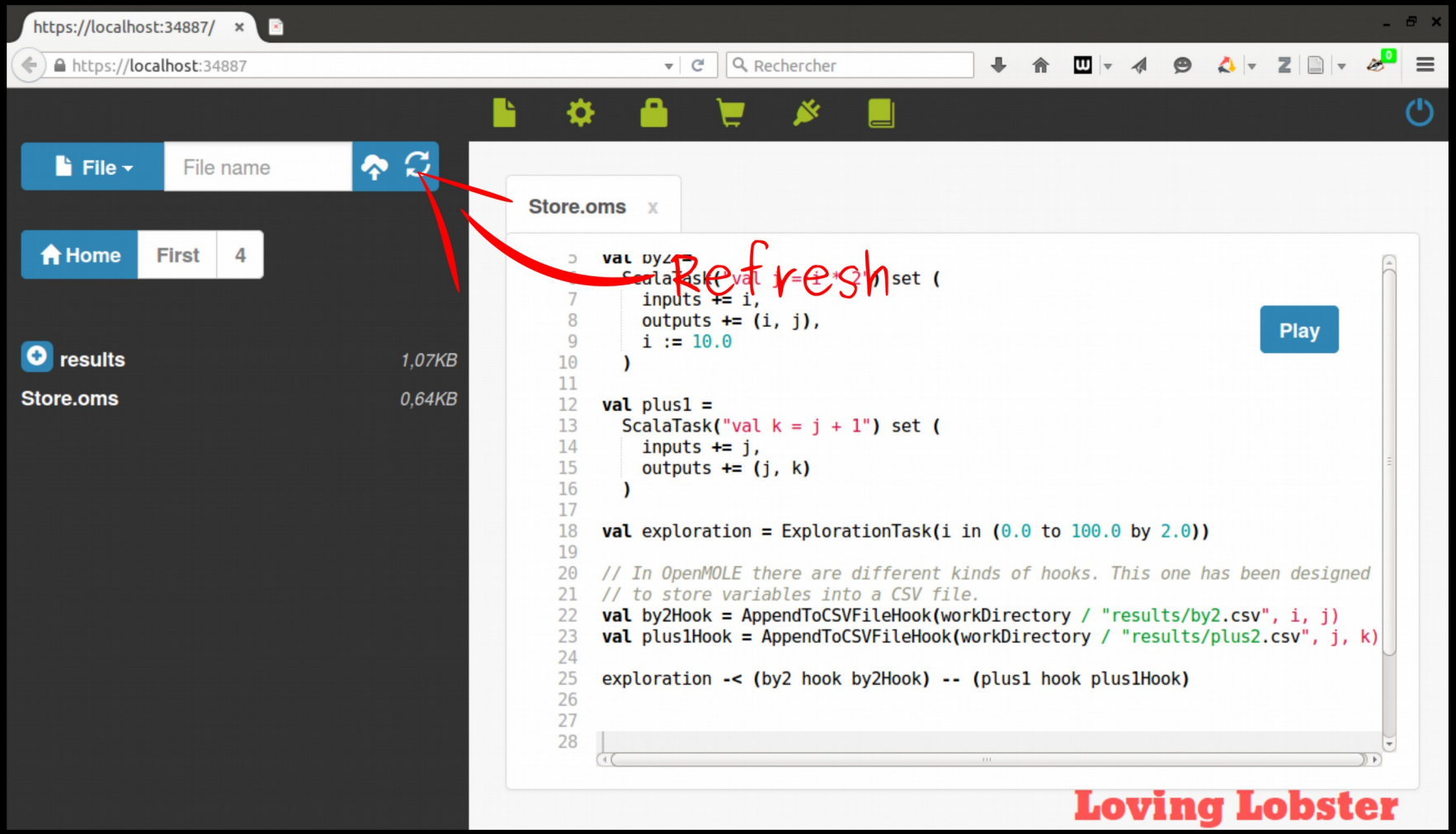
"workDirectory" is the directory of the script.

Store results in files

```
25  exploration -< (by2 hook by2Hook) -- (plus1 hook plus1Hook)
```

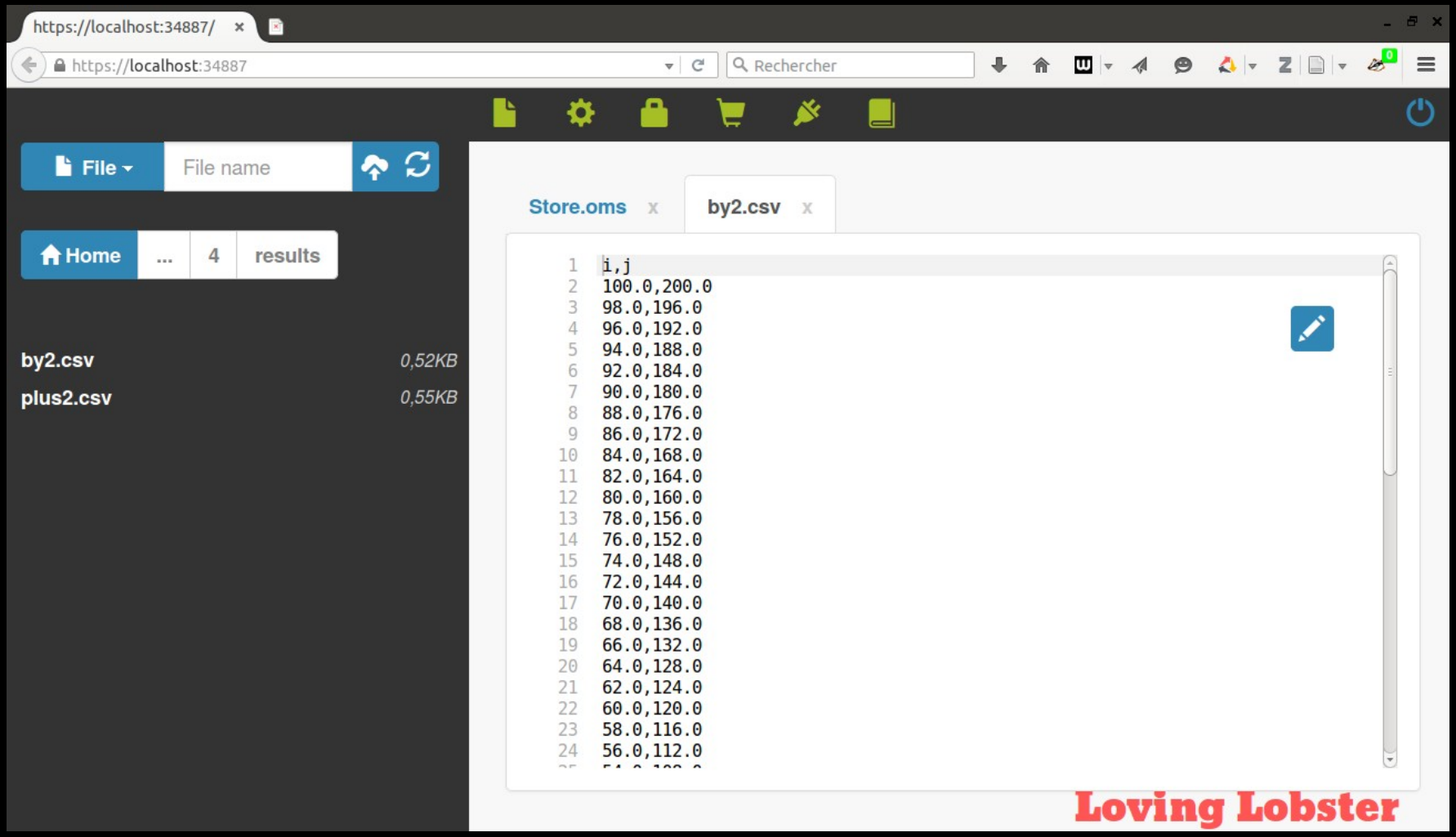
The hooks are hooked on the tasks.

Store results in files



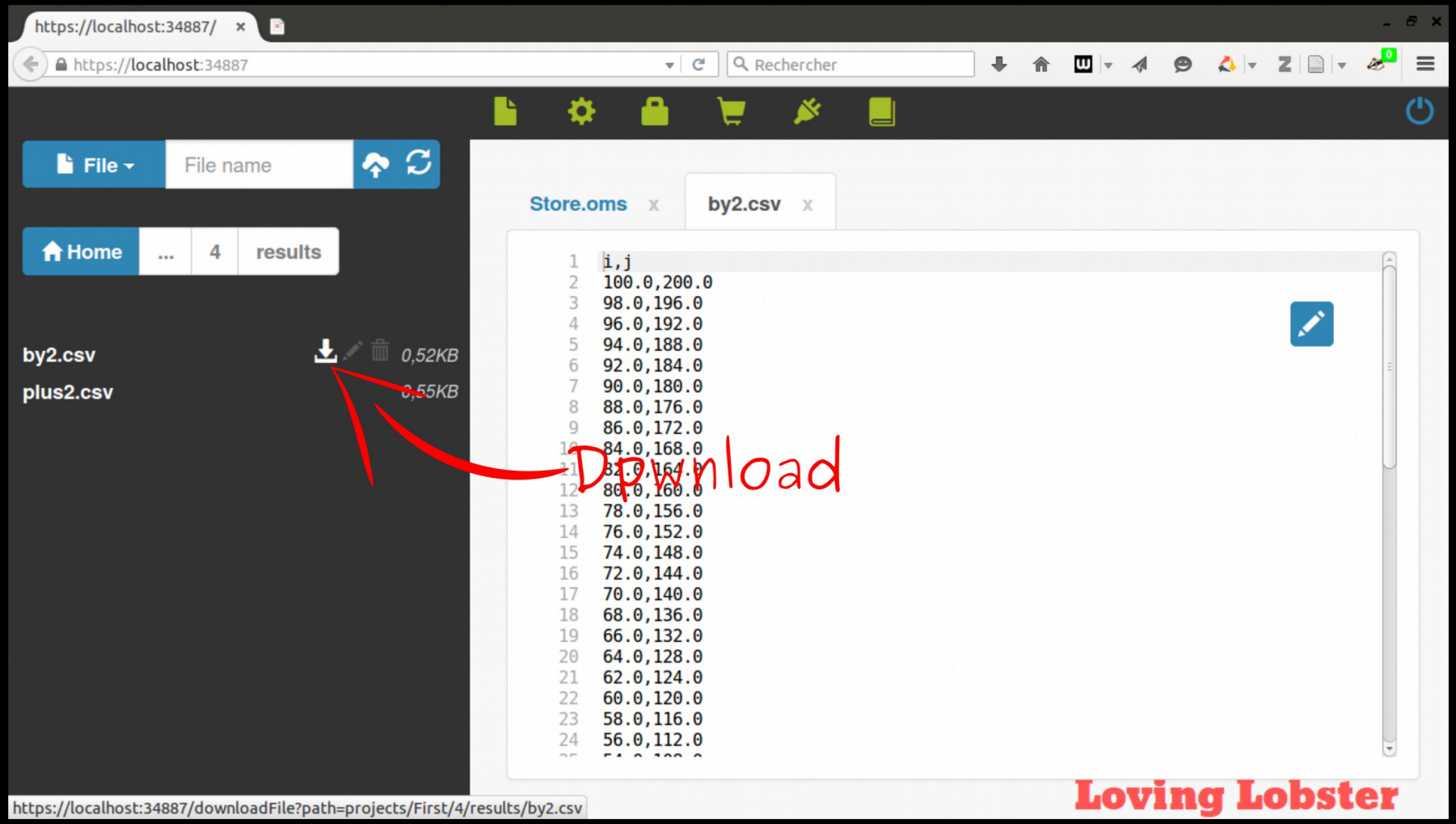
Run it and refresh.

Store results in files



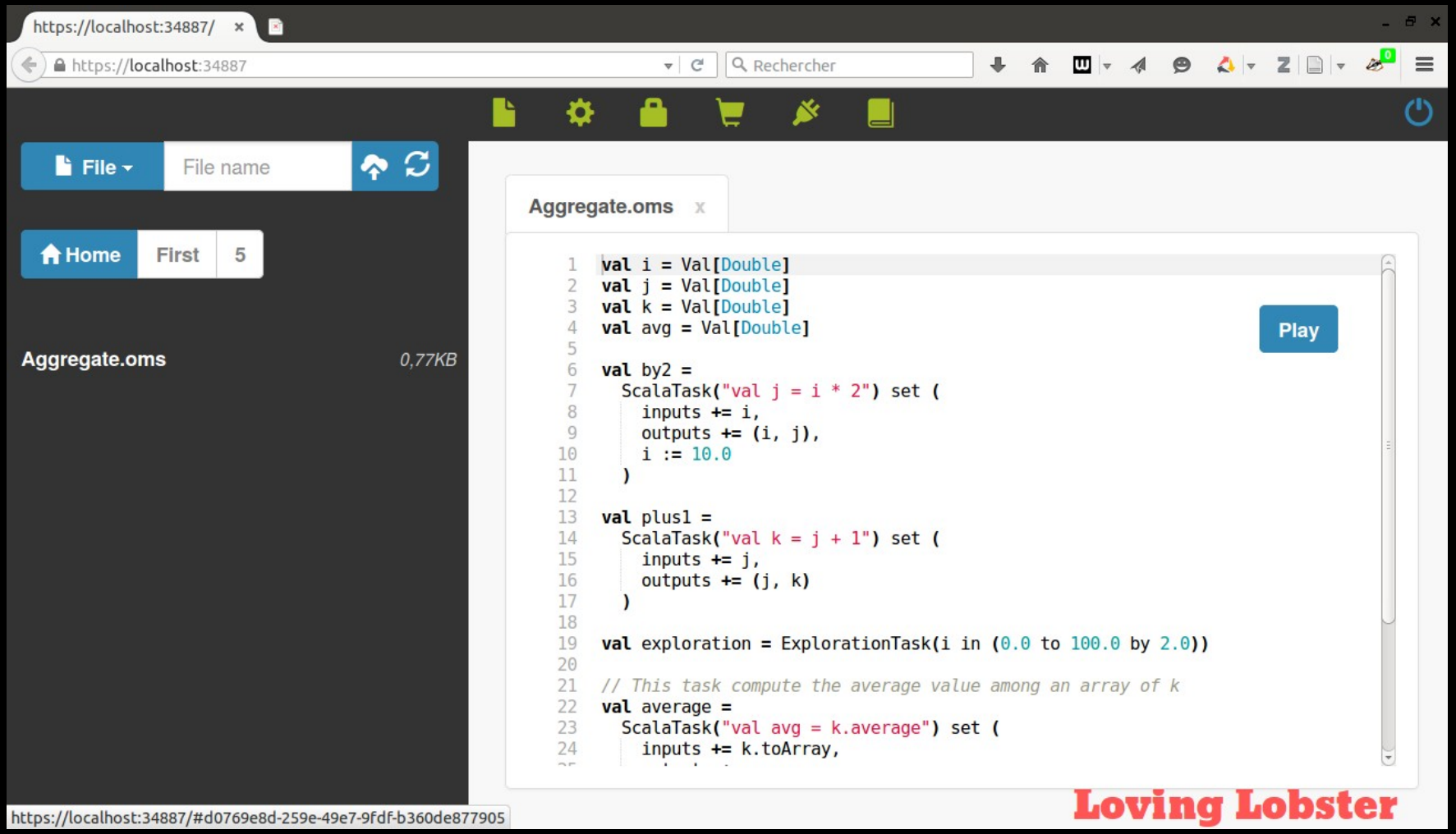
Results have been stored in the "results" folder.

Store results in files



You can download it.

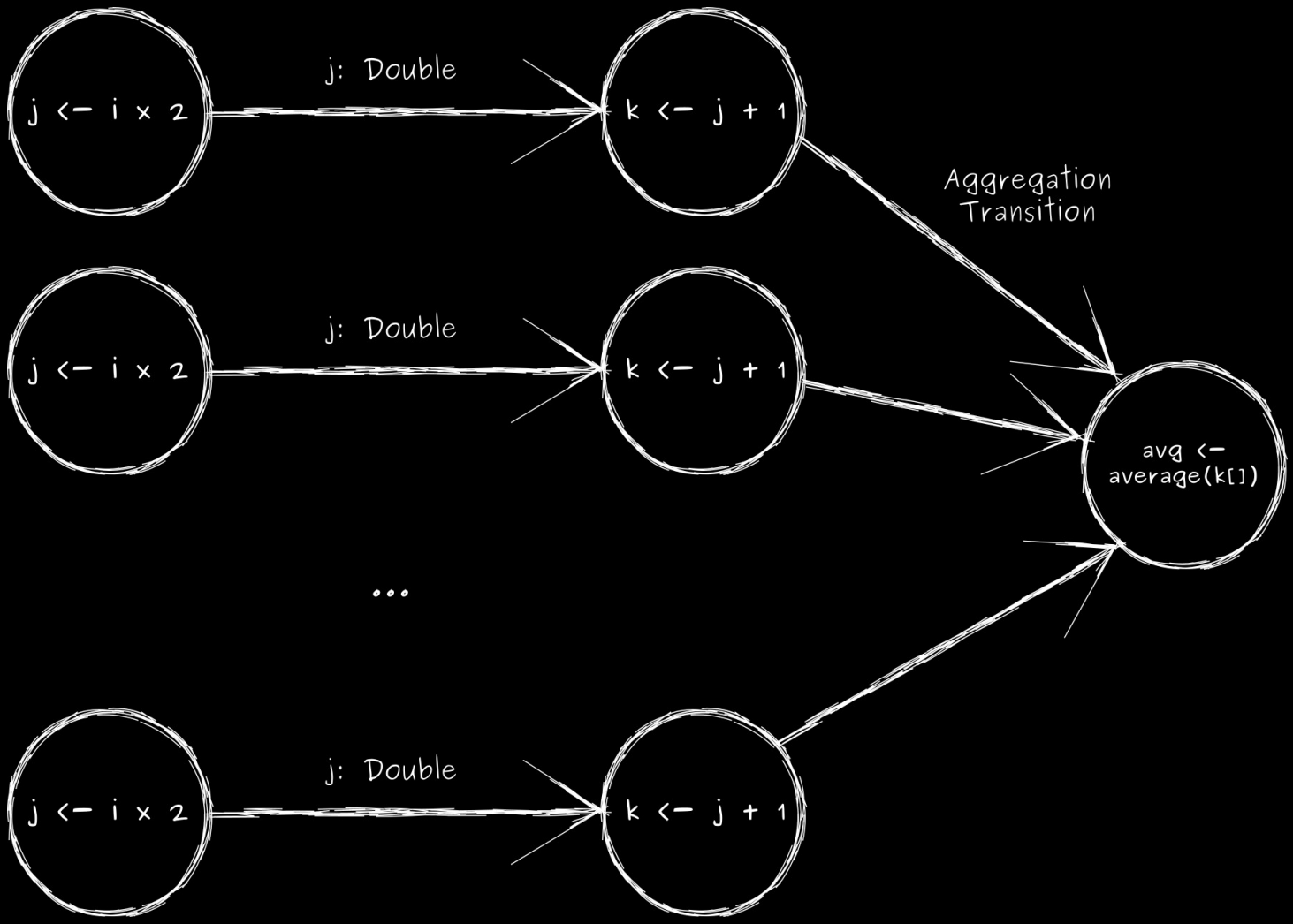
Aggregate



Go to the directory "5"

Click on "Aggregate.oms"

Aggregate



Gathers the results

Aggregate

```
21  // This task compute the average value among an array of k
22  val average =
23    ScalaTask("val avg = k.average") set (
24      inputs += k.toArray,
25      outputs += avg
26    )
```

Computes the average over the values of k.

Takes the input "k: Array[Double]".

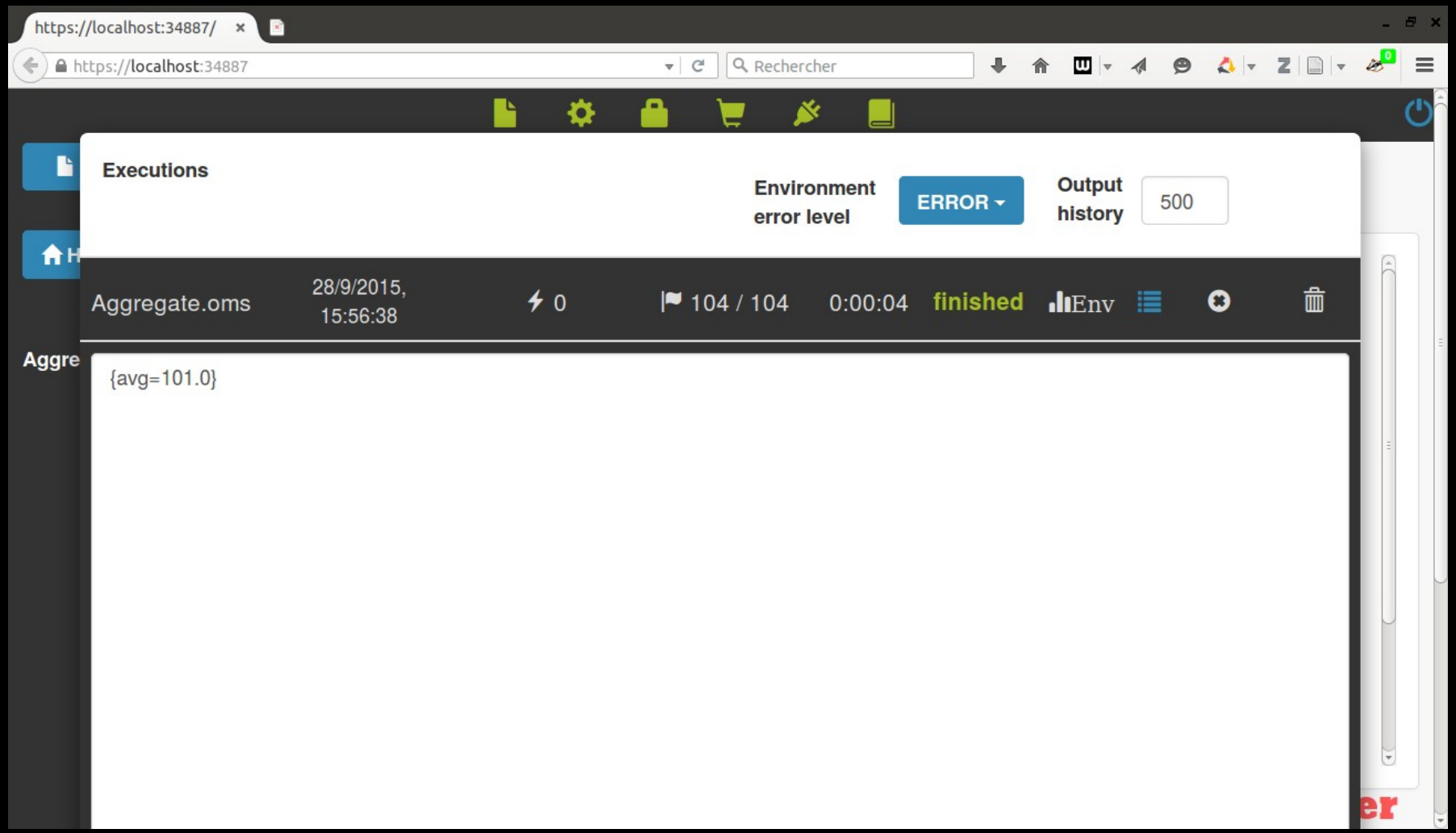
Produces "avg: Double".

Aggregate

```
28  // The ">-" transition waits for the completion of all execution streams.
29  // Then it generates an array of values containing all the values of "k"
30  // generated by the executions of the task "plus1" and launches "average".
31  exploration -< by2 -- plus1 >- (average hook ToStringHook())
```

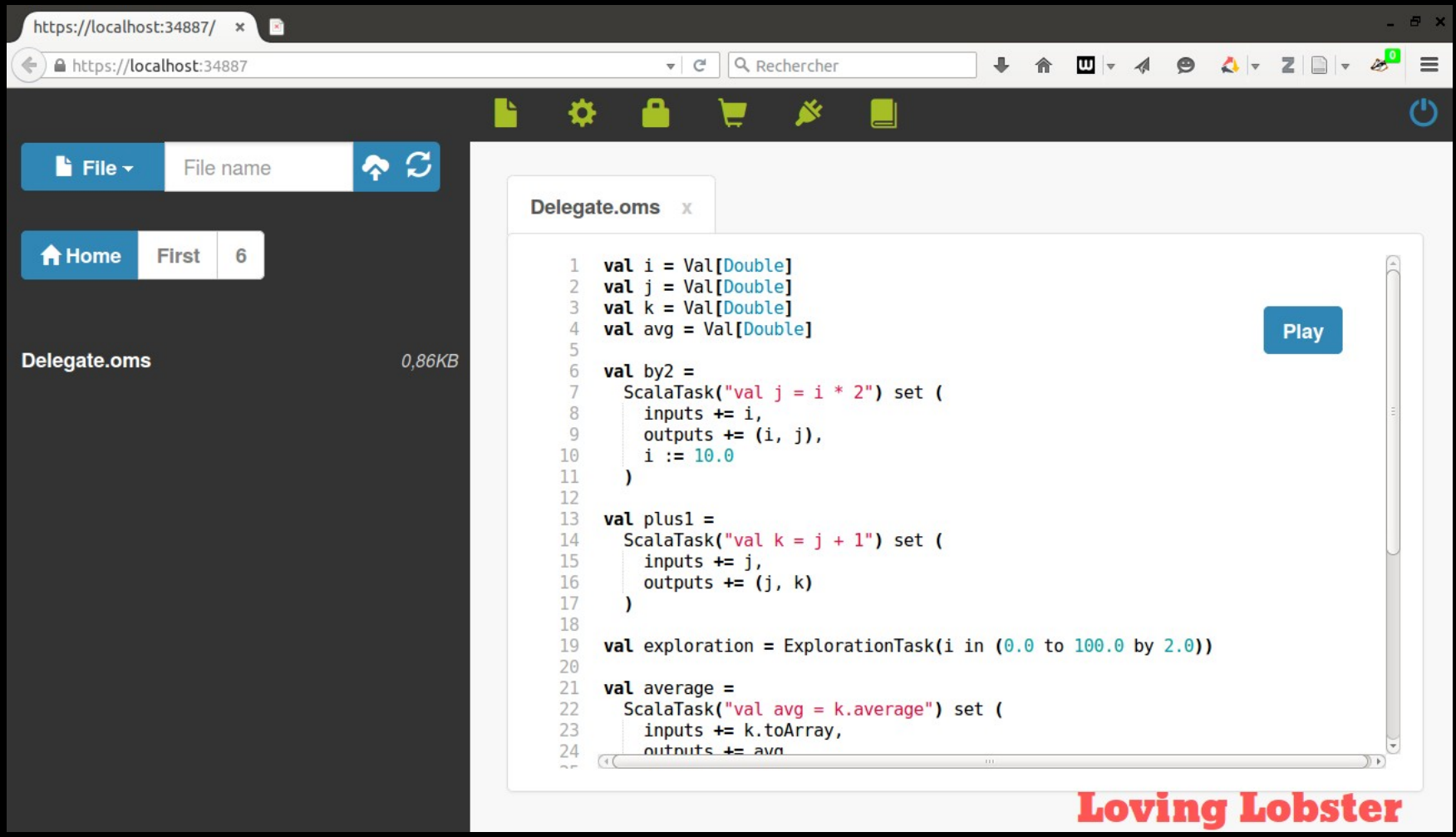
The aggregation transition is noted ">-".
It gathers the results among the execution streams produced by the matching "-<".

Aggregate



Run it.

Delegate the execution



Go to the directory "6"

Click on "Delegate.oms"

Delegate the execution

```
27  // Run 4 parallel processes on the local computer
28  val env = LocalEnvironment(4)
```

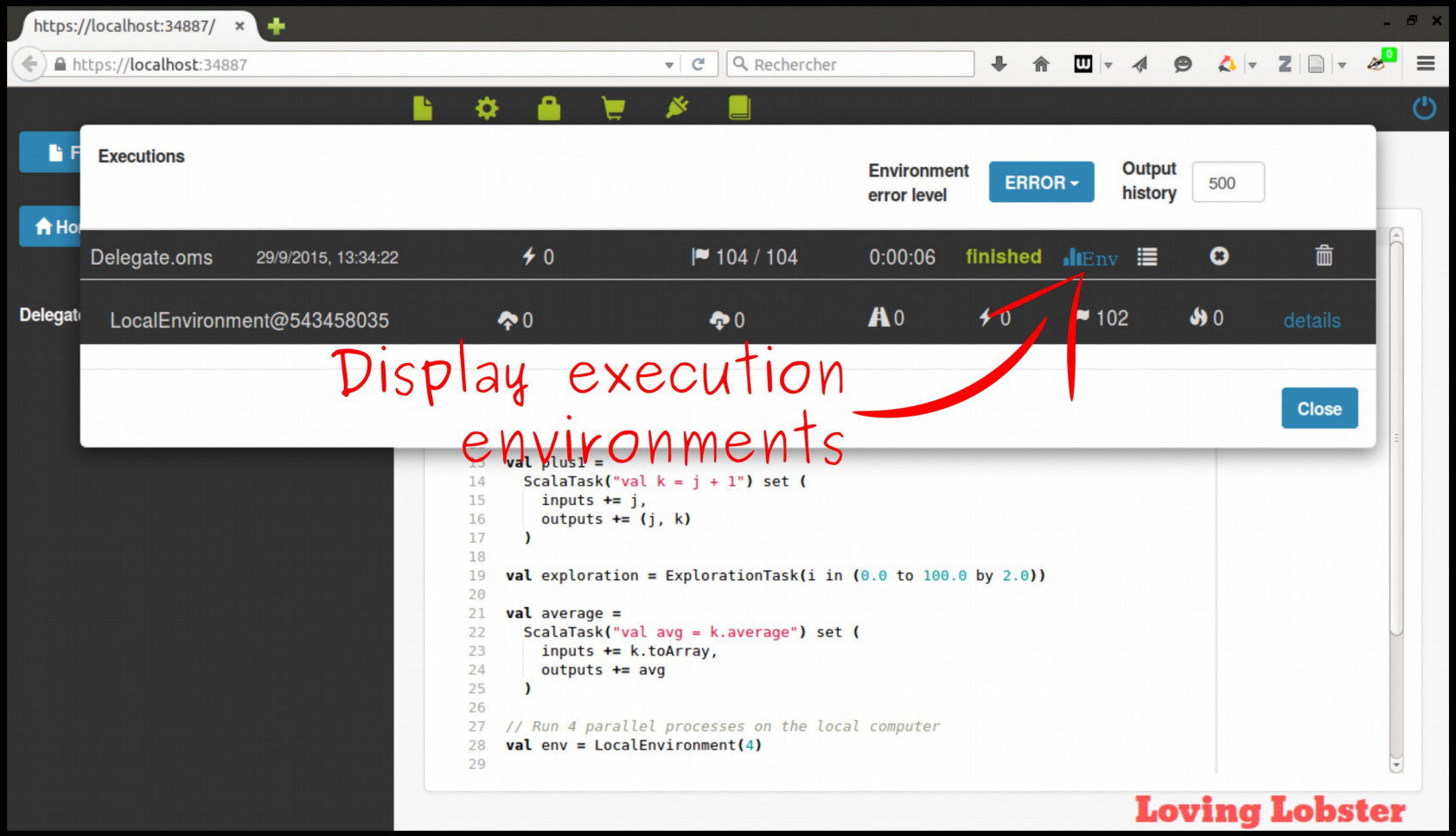
Declare an execution environment:
4 concurrent local executions

Delegate the execution

```
35  // The on keyword is used to delgate tasks to execution environmens
36  exploration -< (by2 on env) -- (plus1 on env) >- (average hook ToStringHook())
```

Use the "on" keyword to delegate the task executions.

Delegate the execution



Run it!

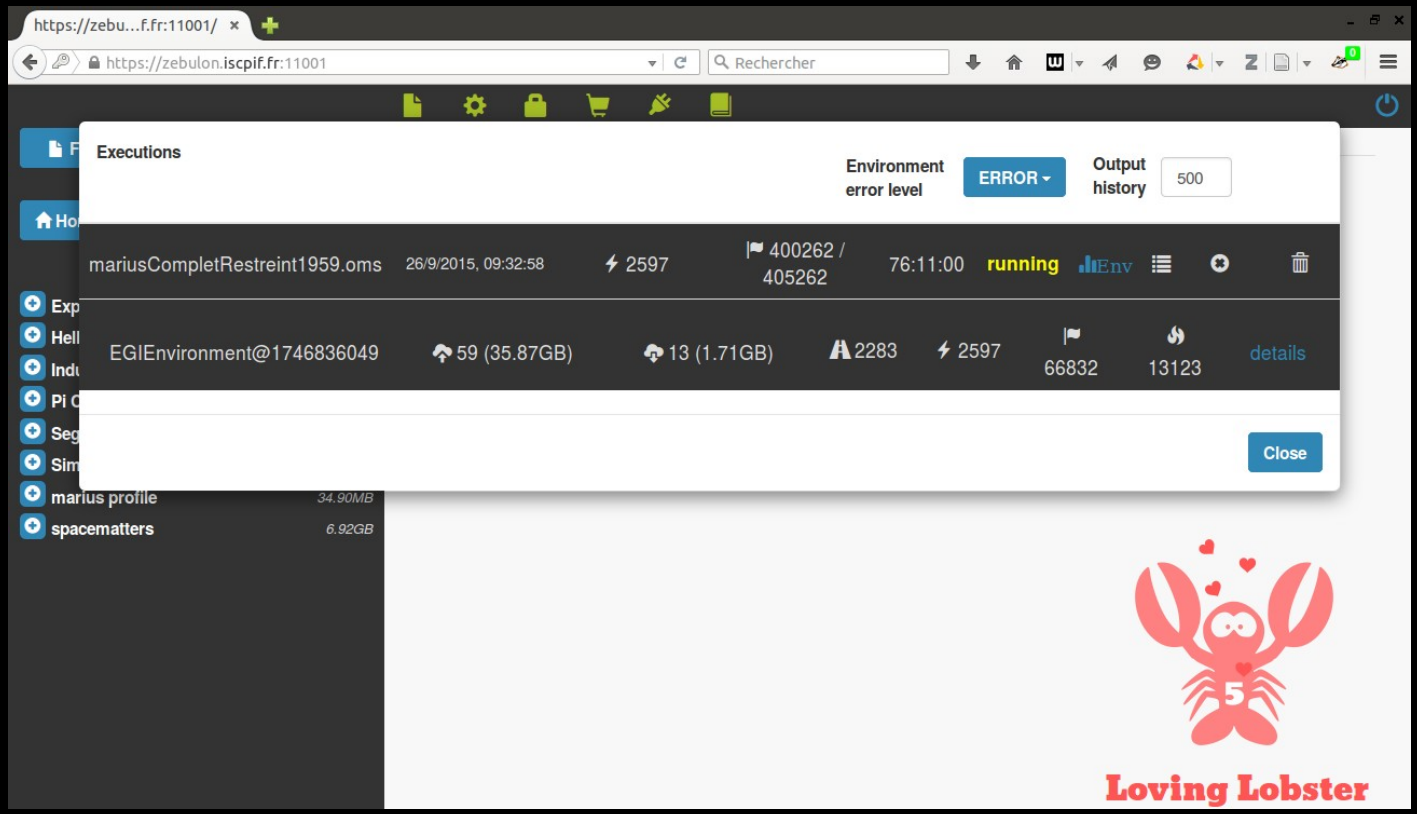
Delegate the execution

Many execution environments are supported:

- most cluster systems
- the EGI Grid
- multi-core servers
- desktop-grid
- … cloud is planed

If your favorite environment is not supported yet, please contact us.

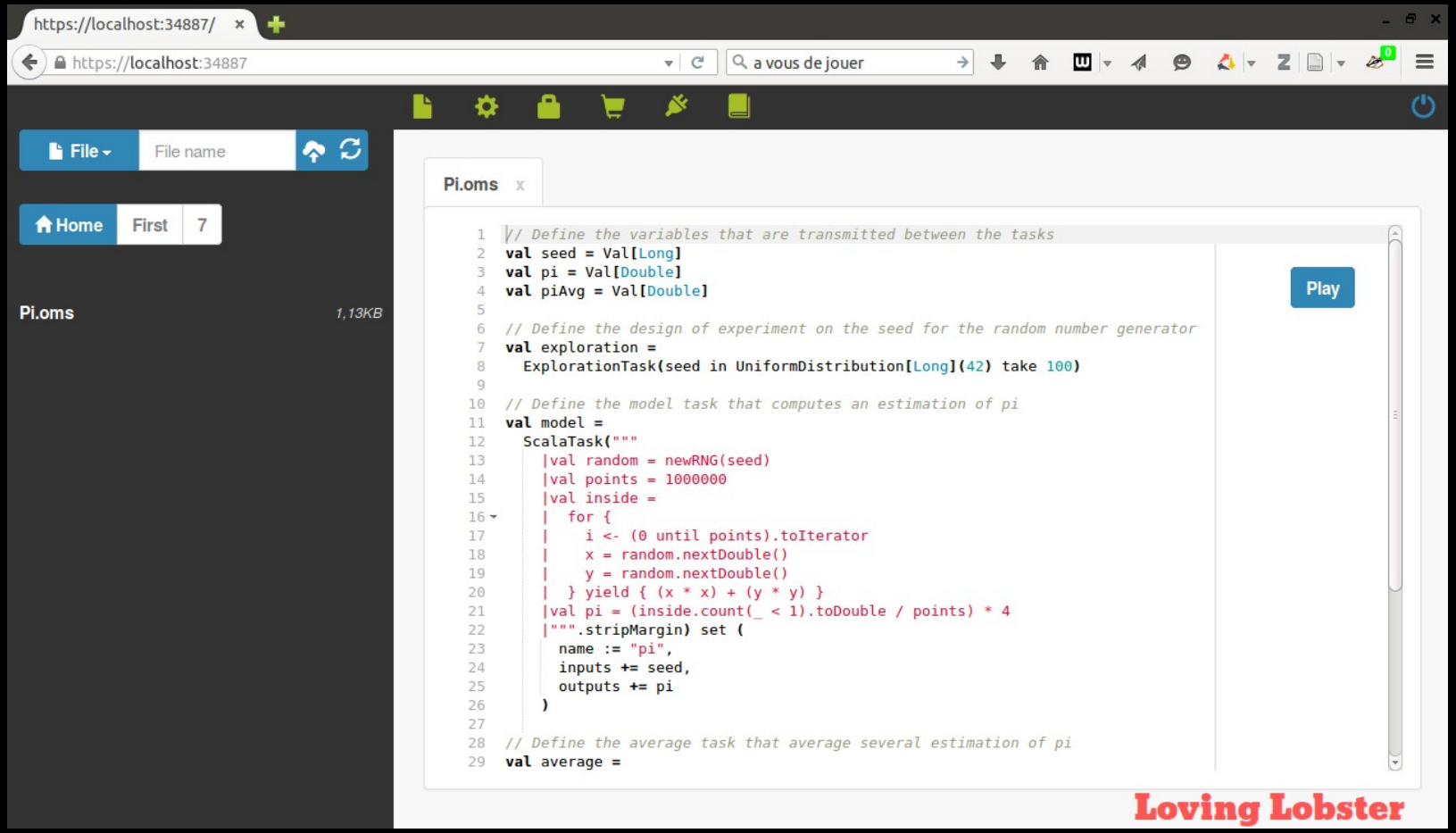
Delegate the execution



OpenMOLE scales!

Millions of jobs, +100GB of data

And now, play on!




```
// Define the variables that are transmitted between the tasks
val seed = Val[Long]
val pi = Val[Double]
val piAvg = Val[Double]

// Define the design of experiment on the seed for the random number generator
val exploration =
  ExplorationTask(seed in UniformDistribution[Long](42) take 100)

// Define the model task that computes an estimation of pi
val model =
  ScalaTask("""
    |val random = newRNG(seed)
    |val points = 1000000
    |val inside =
    |  for {
    |    i <- (0 until points).toIterator
    |    x = random.nextDouble()
    |    y = random.nextDouble()
    |  } yield { (x * x) + (y * y) }
    |val pi = (inside.count(_ < 1).toDouble / points) * 4
    |""".stripMargin) set (
      name := "pi",
      inputs += seed,
      outputs += pi
    )

// Define the average task that average several estimation of pi
val average =
```

Go to the directory "7"

Click on "Pi.oms"

And now, play on!

Pi: computes an approximation of Pi using the Monte-Carlo method

Exploration: generates seeds for the pseudo-random numbers generator of Pi

Average: compute the average among several realizations of Pi

And now, play on!

Objective 1: distribute the executions of the task "Pi"

Objective 2: aggregate the values computed by "Pi" and display the average

Coming next:

How to distribute a native (Linux) application (R, Scilab, C++, Python…) to thousands of computers.