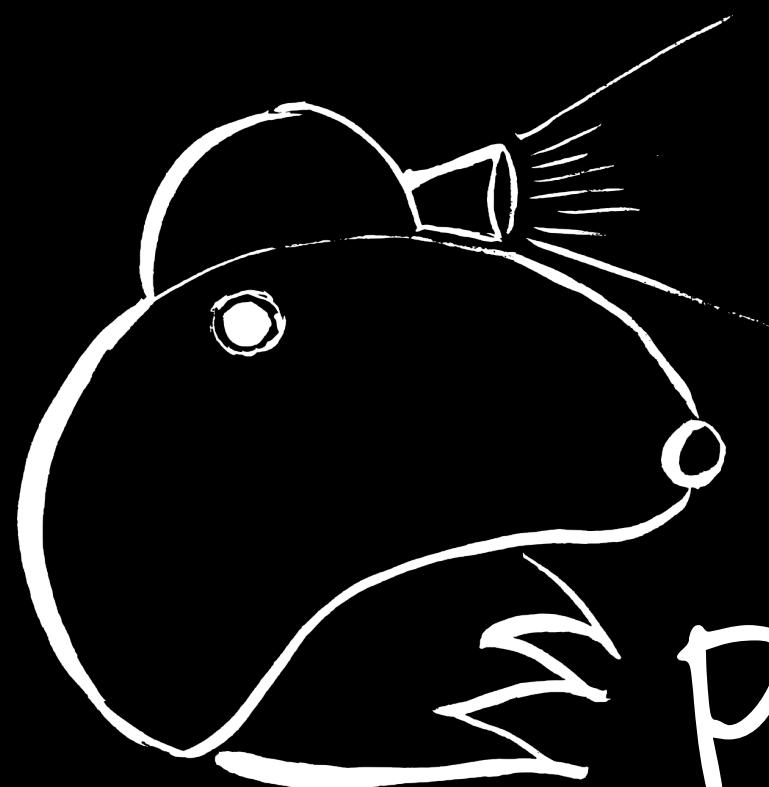


First steps with



penMOLE

# Prerequisite

Make sure you have the Java runtime environment 7 or 8

```
reuillon@simplet: /home/reuillon
[reuillon:~] 25s $ java -version
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jay
atanaag.jar
openjdk version "1.8.0_66-internal"
OpenJDK Runtime Environment (build 1.8.0_66-internal-b01)
OpenJDK 64-Bit Server VM (build 25.66-b01, mixed mode)
[reuillon:~] 3s $
```

# Prerequisite

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

# Download

Go to: [www.openmole.org](http://www.openmole.org)

Go to: Getting Started.

The screenshot shows a web browser window with the OpenMOLE logo at the top. Below the logo, there are three navigation links: "Getting Started", "Documentation", and "Who are we?". A prominent green button labeled "Download OpenMOLE 5.4" is centered below these links. A red arrow points from the bottom right towards this button. Below the button, a small note states: "Loving Lobster (version 5.4) was released on September 30, 2015." The main content area is titled "Launch" and contains instructions for Java installation and launching the application. Another section titled "First workflow" provides a brief introduction to workflow design.

**Getting Started Documentation Who are we?**

**Download OpenMOLE 5.4**

Loving Lobster (version 5.4) was released on September 30, 2015.

## Launch

OpenMOLE requires that Java version 7 or above is installed and available on your computer. Then just extract the archive, and you're done! OpenMOLE is installed and works out of the box!

Once installed you can launch OpenMOLE by executing the `openmole` file in the installation directory (it is called `openmole.bat` for windozers). It will bring up the OpenMOLE **application** that runs in your web browser (OpenMOLE supports Firefox and Chrome).

## First workflow

Let's design a workflow that explores one variable multiplies it by 2 and then displays each result. The computation is executed in a multi-threaded environment on the local machine.

In the OpenMOLE interface, first create a file named `example.oms` (the file extension is important). Open it and write the following workflow:

# Launch

Run:

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

# Insecure connection

Connexion non certifiée

https://localhost:36093

Rechercher

Connexion non certifiée

Cette connexion n'est pas certifiée

Vous avez demandé à Firefox de se connecter de manière sécurisée à **localhost:36093**, mais nous ne pouvons pas confirmer que votre connexion est sécurisée.

Normalement, lorsque vous essayez de vous connecter de manière sécurisée, les sites présentent une identification certifiée pour prouver que vous vous trouvez à la bonne adresse. Cependant, l'identité de ce site ne peut pas être vérifiée.

Que dois-je faire ?

Si vous vous connectez habituellement à ce site sans problème, cette erreur peut signifier que quelqu'un essaie d'usurper l'identité de ce site et vous ne devriez pas continuer.

Sortir d'ici !

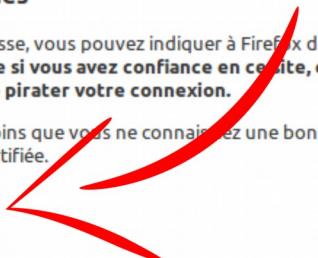
Détails techniques

Je comprends les risques

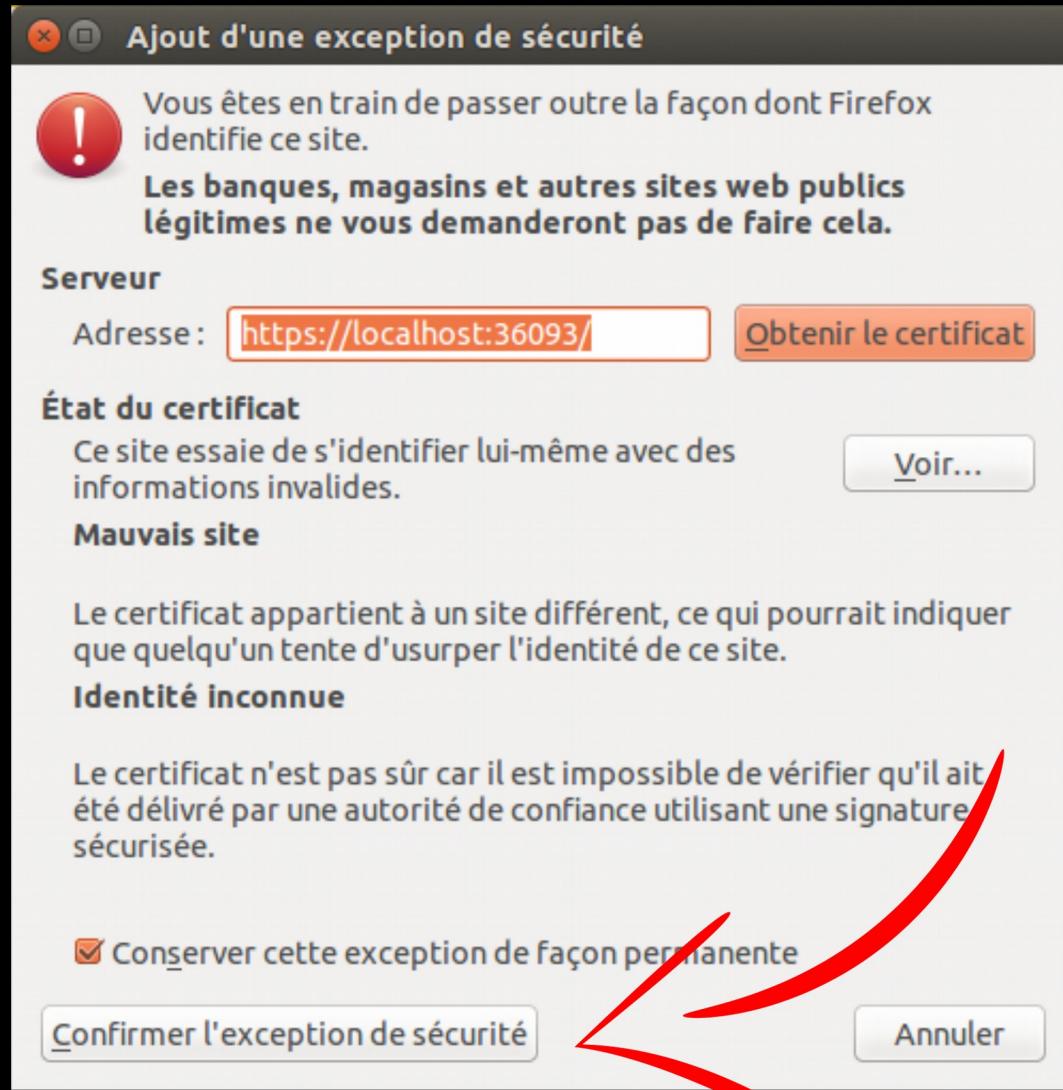
Si vous comprenez ce qui se passe, vous pouvez indiquer à Firefox de commencer à faire confiance à l'identification de ce site. **Même si vous avez confiance en ce site, cette erreur pourrait signifier que quelqu'un est en train de pirater votre connexion.**

N'ajoutez pas d'exception à moins que vous ne connaissez une bonne raison pour laquelle ce site n'utilise pas d'identification certifiée.

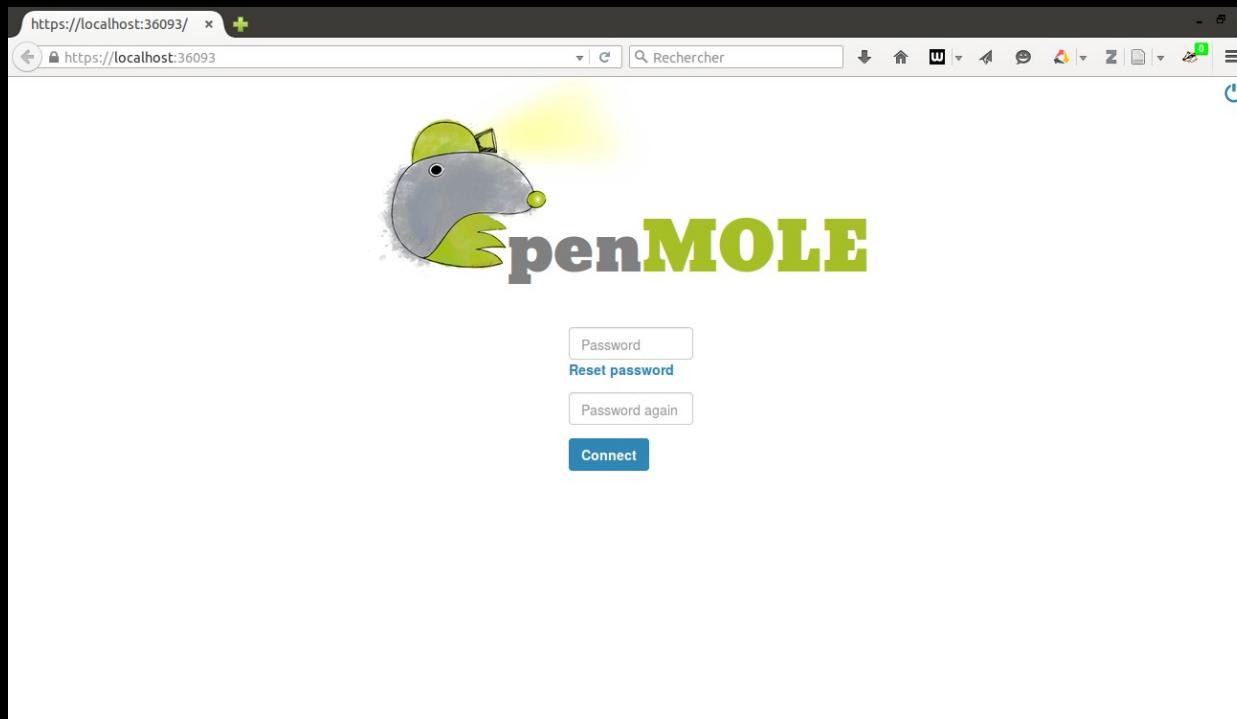
Ajouter une exception...



# Insecure connection



# OpenMOLE password



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

# Welcome to OpenMOLE

The screenshot shows the OpenMOLE web interface at <https://localhost:34887/>. The interface has a dark theme with a top navigation bar containing various icons. A sidebar on the left includes a 'File' dropdown, a 'File name' input field with a blue 'Upload' button, and a 'Home' button. The main area displays the text 'Create a first OpenMOLE script (.oml)'. Red annotations have been added to highlight specific features: a double-headed arrow between the 'File' dropdown and the 'Upload' button is labeled 'Upload files'; a single-headed arrow pointing from the 'File' dropdown towards the main area is labeled 'Create new files'; and a single-headed arrow pointing from the left sidebar towards the main area is labeled 'File browser'.

Upload files

Create new files

File browser



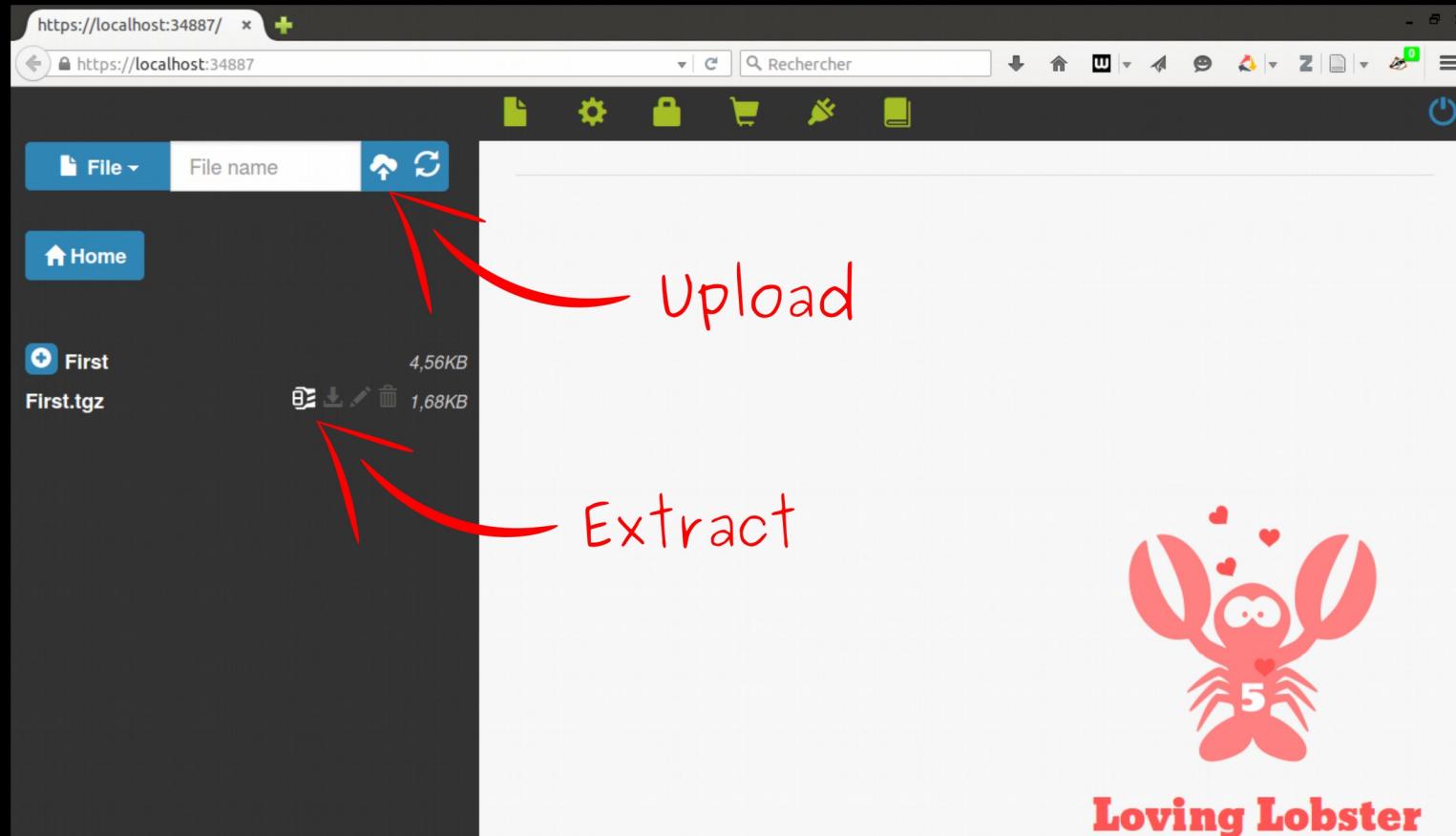
**Loving Lobster**

# First workflow

Go to: [www.openmole.org/files/csdcc](http://www.openmole.org/files/csdcc)

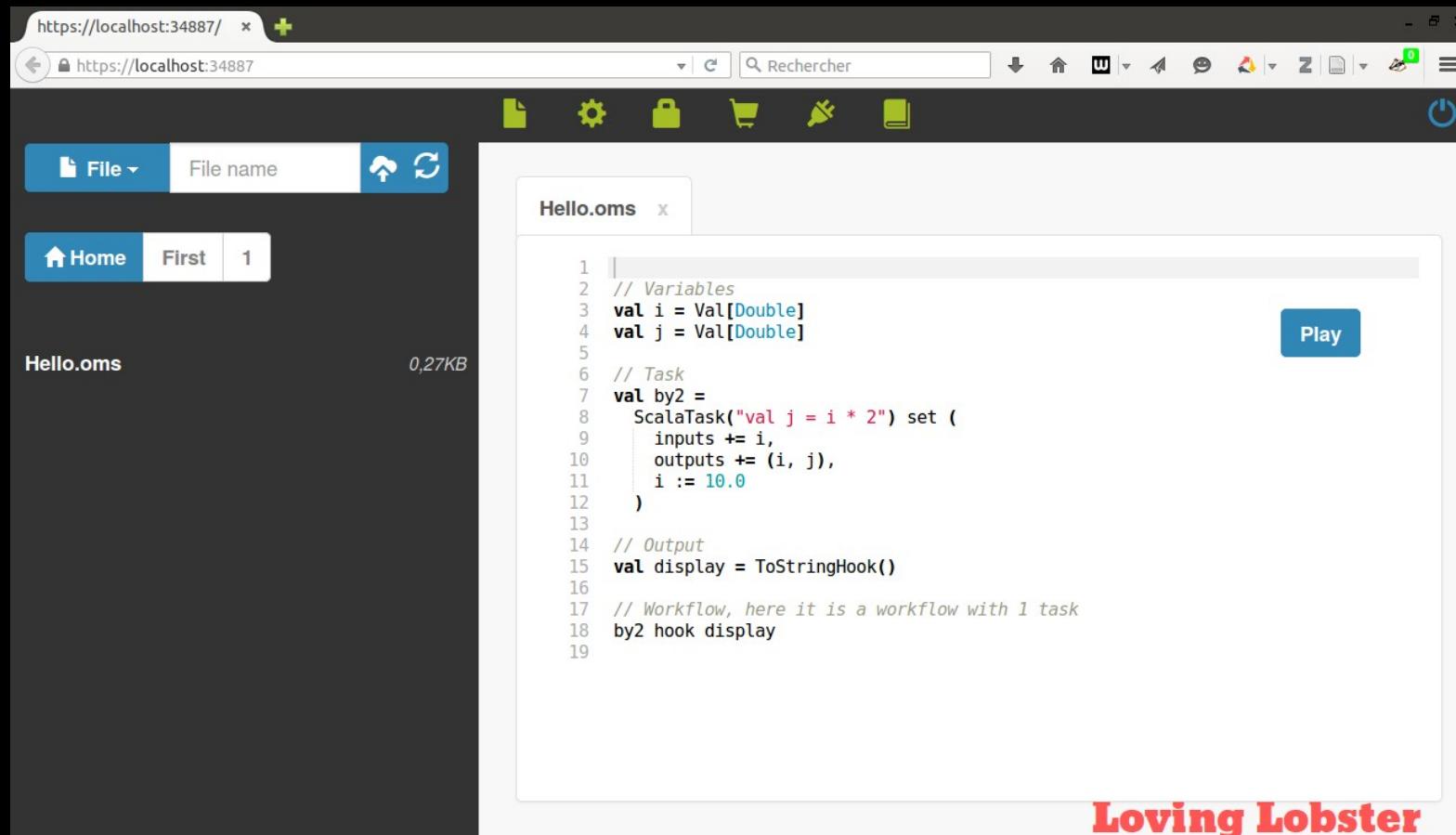
Download First.tgz

# First workflow



Upload the archive  
Extract it

# First workflow

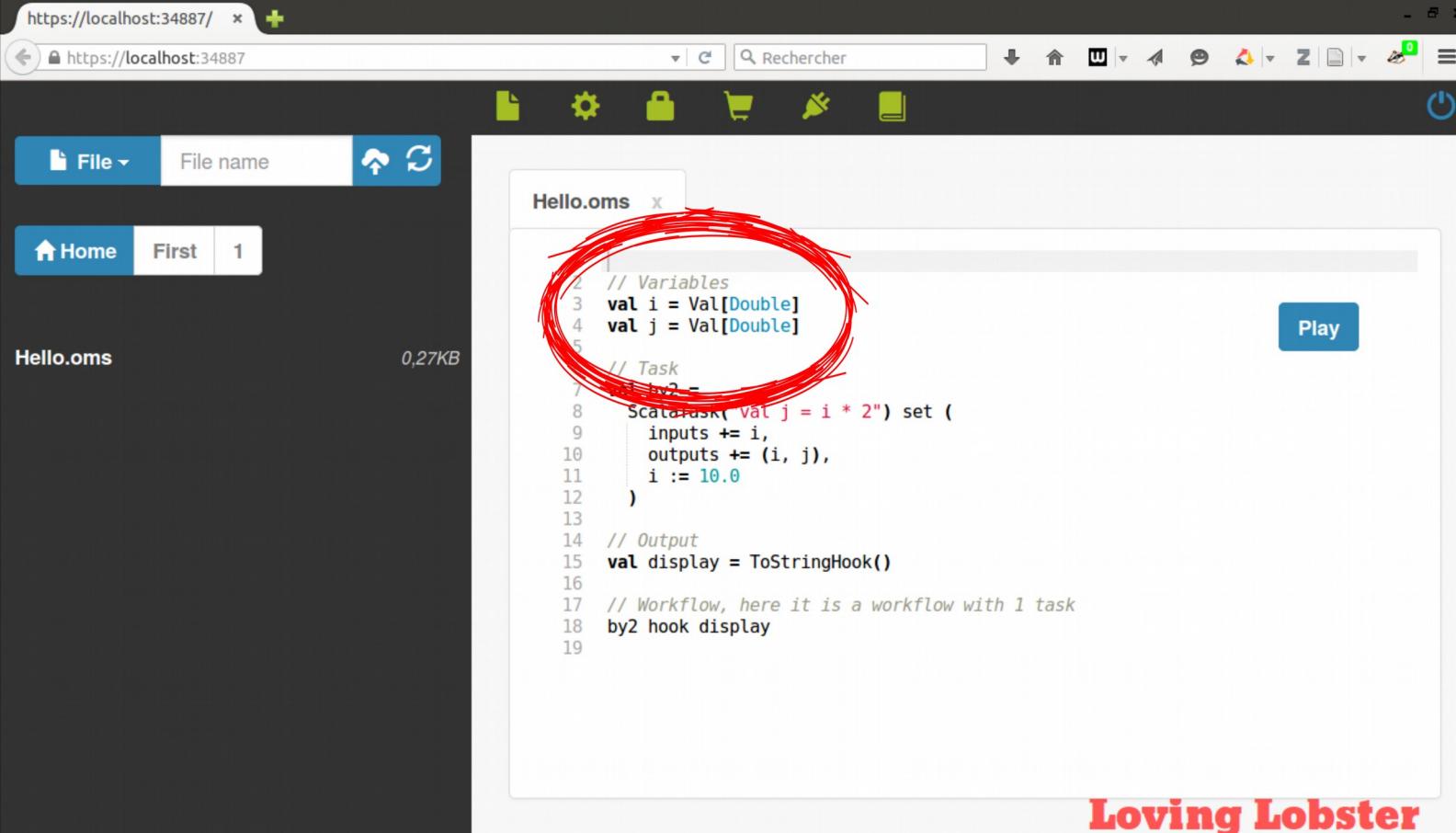


Click on the folder "First"

Click on the folder "1"

Click on the file "Hello.oms"

# First workflow



The screenshot shows a web-based interface for managing workflows. At the top, there's a browser header with the URL <https://localhost:34887/>. Below it is a toolbar with various icons. The main area has a file manager on the left and a code editor on the right.

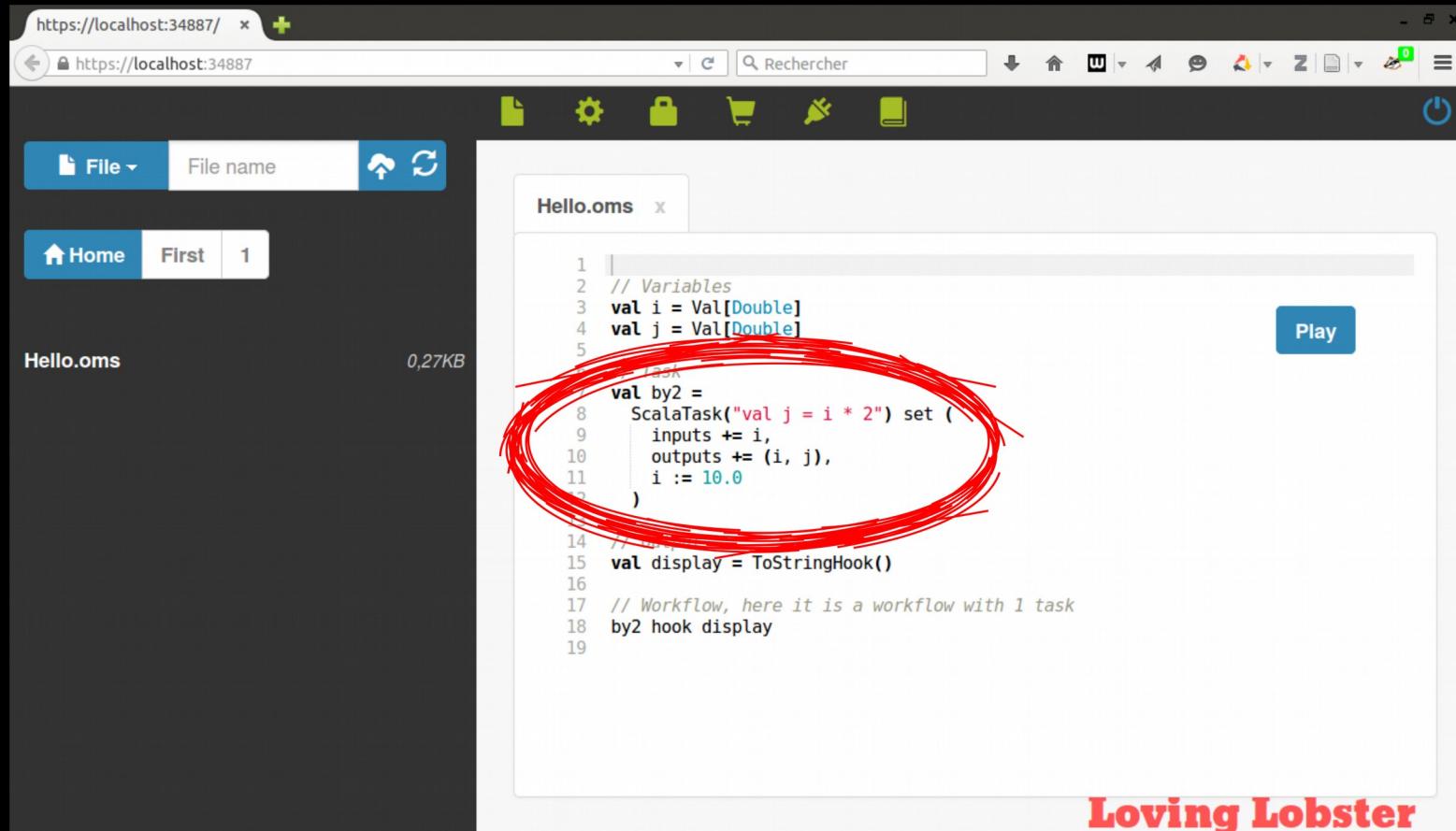
The code editor window is titled "Hello.oms". It contains the following Scala code:

```
1 // Variables
2 val i = Val[Double]
3 val j = Val[Double]
4
5 // Task
6 i >= j ->
7   Scaladask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13
14 // Output
15 val display = ToStringHook()
16
17 // Workflow, here it is a workflow with 1 task
18 by2 hook display
19
```

A large red circle highlights the first two lines of code, which declare typed variables `i` and `j`. To the right of the code editor is a "Play" button. At the bottom right of the entire interface, the text "Loving Lobster" is displayed in red.

Express the dataflow  
Variables are typed

# First workflow



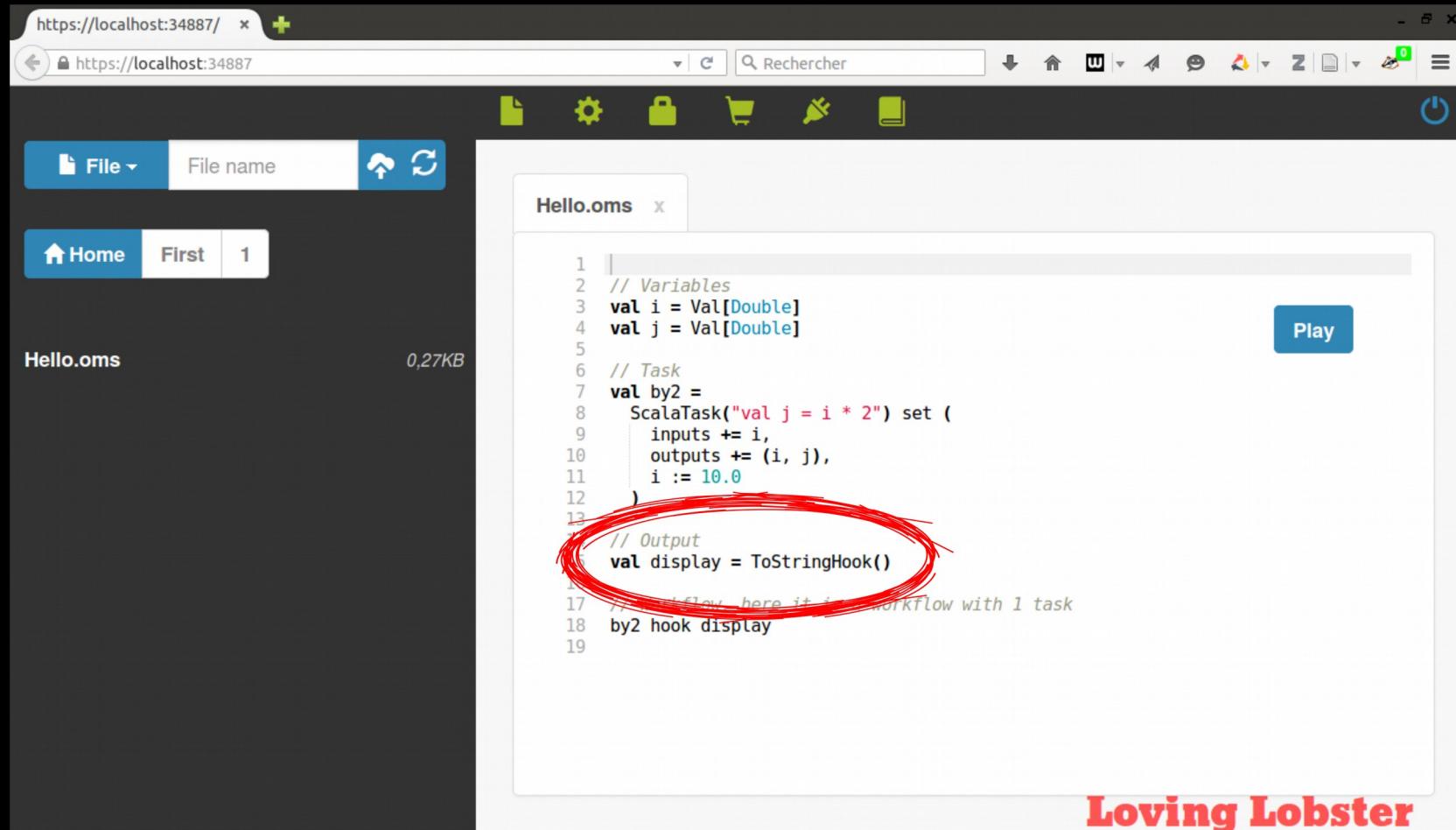
The screenshot shows a web-based Scala code editor interface. The URL is <https://localhost:34887/>. The code editor window is titled "Hello.oms". The code itself is:

```
1 // Variables
2 val i = Val[Double]
3 val j = Val[Double]
4
5 // TASK
6 val by2 =
7   ScalaTask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13 // WORKFLOW
14 val display = ToStringHook()
15
16 // Workflow, here it is a workflow with 1 task
17 by2 hook display
18
19
```

A large red oval highlights the entire ScalaTask definition on lines 6-11. A blue "Play" button is visible on the right side of the code editor. The status bar at the bottom right of the browser window displays "Loving Lobster".

Express computation  
Uses inputs, produces outputs  
(may specify default values)

# First workflow



The screenshot shows a web browser window with the URL <https://localhost:34887/>. The page displays a file named "Hello.oms" with a size of 0,27KB. The file content is a Scala code snippet:

```
1 // Variables
2 val i = Val[Double]
3 val j = Val[Double]
4
5 // Task
6 val by2 =
7   ScalaTask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13 // Output
14 val display = ToStringHook()
15
16 // Workflow here it is a workflow with 1 task
17 by2 hook display
18
19
```

A red oval highlights the line `val display = ToStringHook()`. A blue "Play" button is visible on the right side of the code editor.

Loving Lobster

Express output operations  
Always runs locally

# First workflow

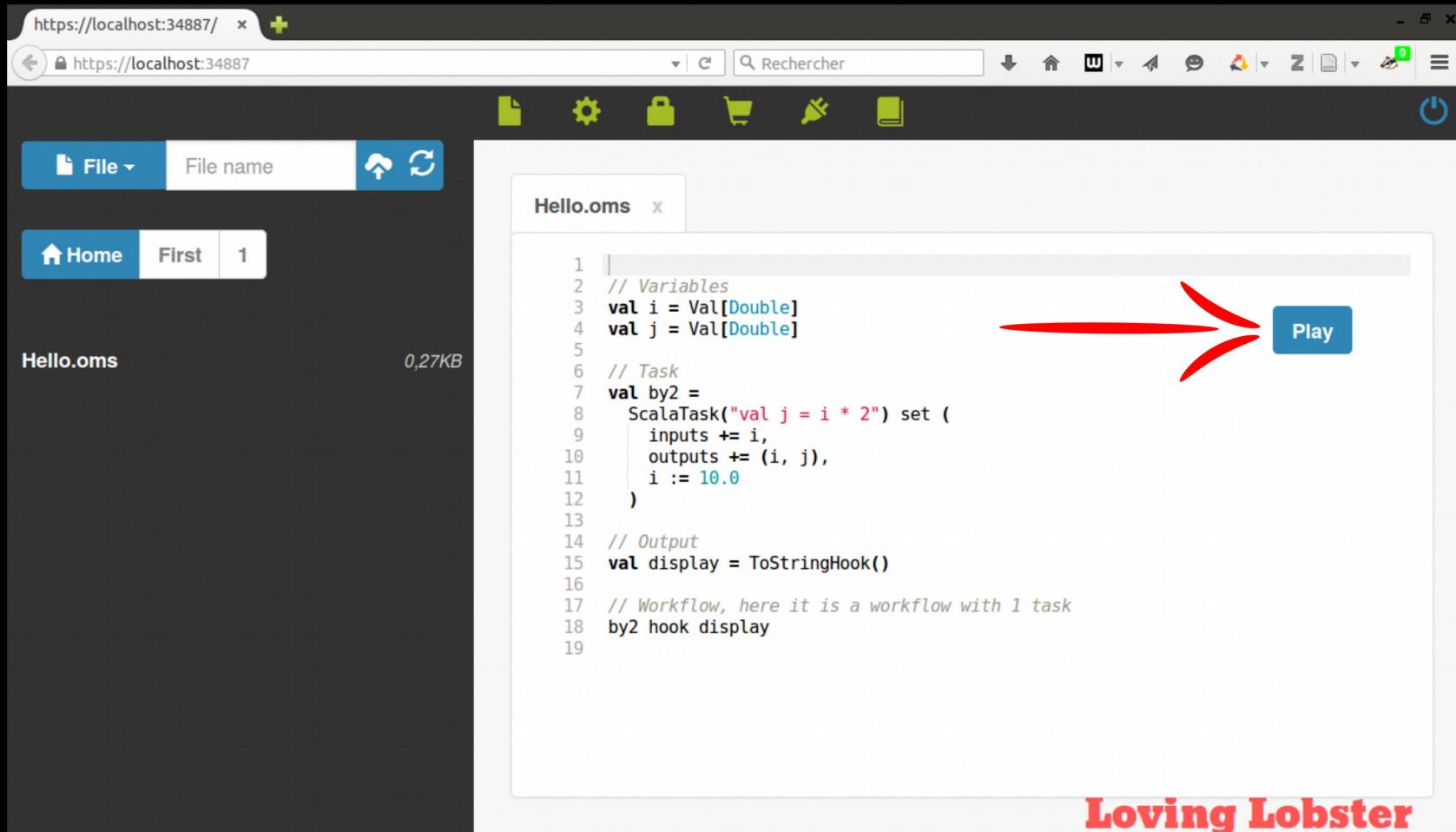
The screenshot shows a web browser window with a dark theme. The address bar shows `https://localhost:34887/`. The main content area displays a file named `Hello.oms` with a size of `0,27KB`. The file content is a Scala code snippet:

```
1 // Variables
2 val i = Val[Double]
3 val j = Val[Double]
4
5 // Task
6 val by2 =
7   ScalaTask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13 // Output
14 val display = ToStringHook()
15
16 // Workflow, here it's a workflow with 1 task
17 by2 hook display
18
```

A red circle highlights the line `by2 hook display`. In the bottom right corner of the code editor, there is a blue button labeled `Play`. At the very bottom of the page, there is a red banner with the text **Loving Lobster**.

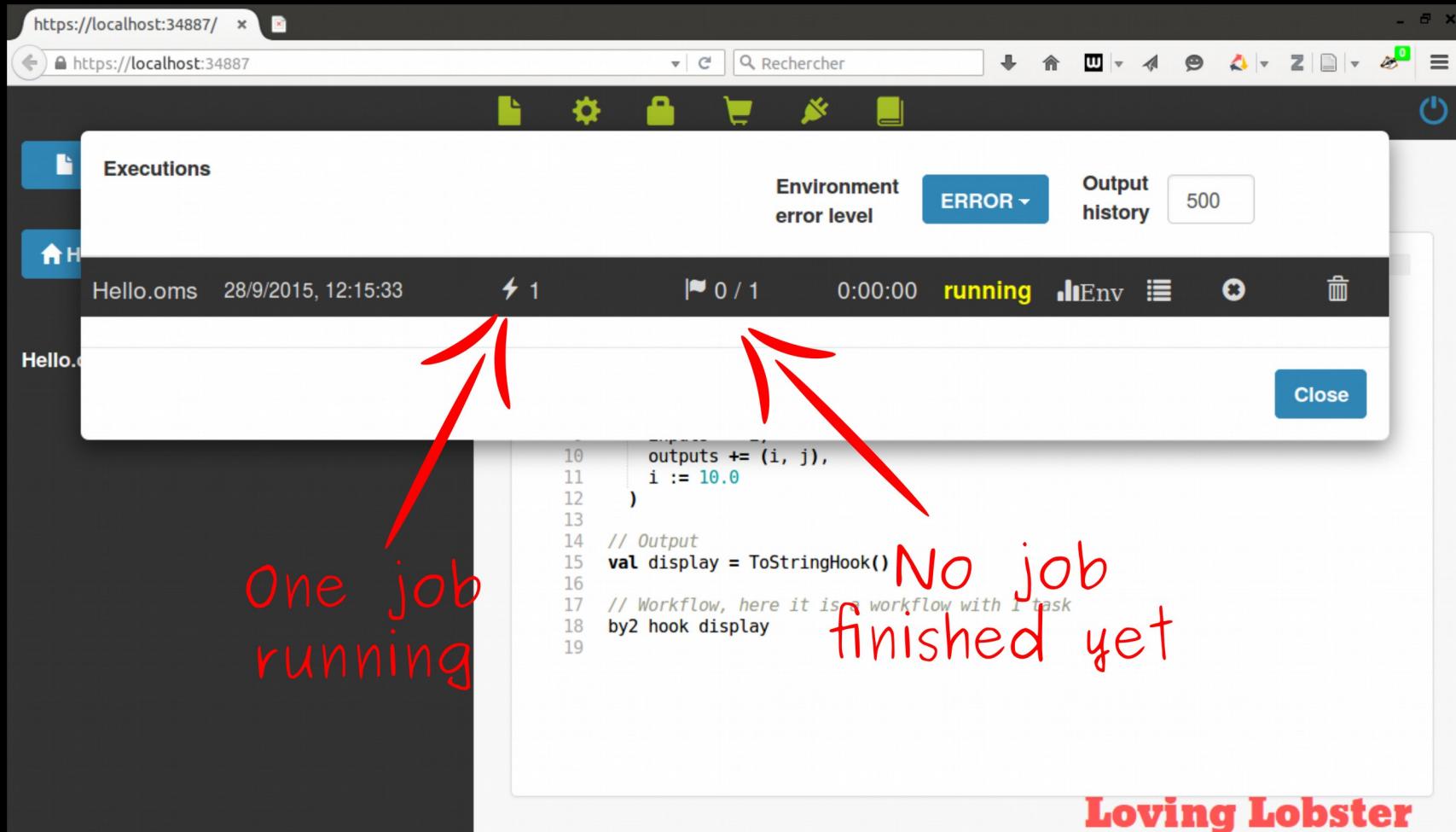
Express workflow composition  
Here: One task + One hook

# First workflow



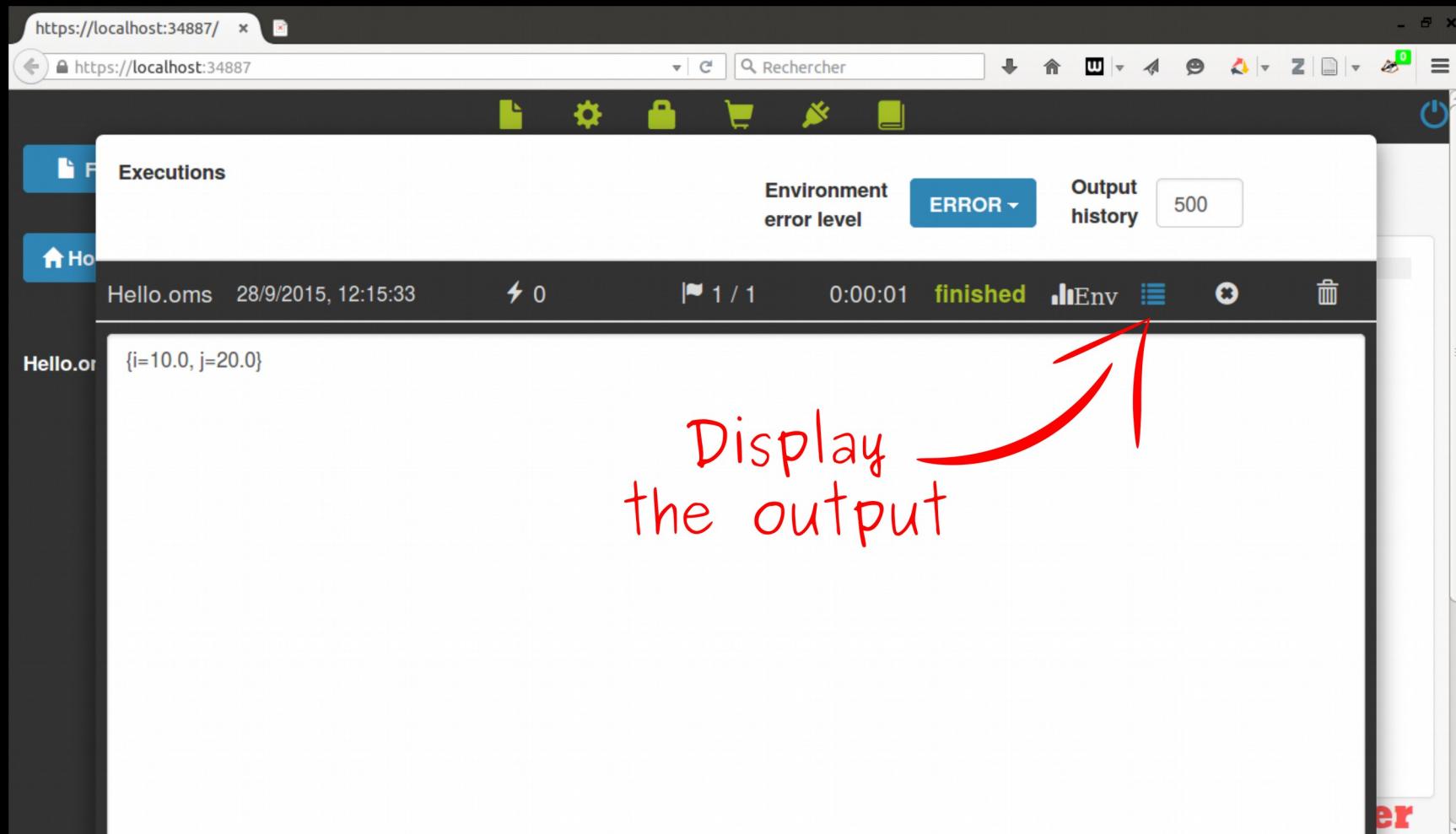
And now lets run!

# First workflow



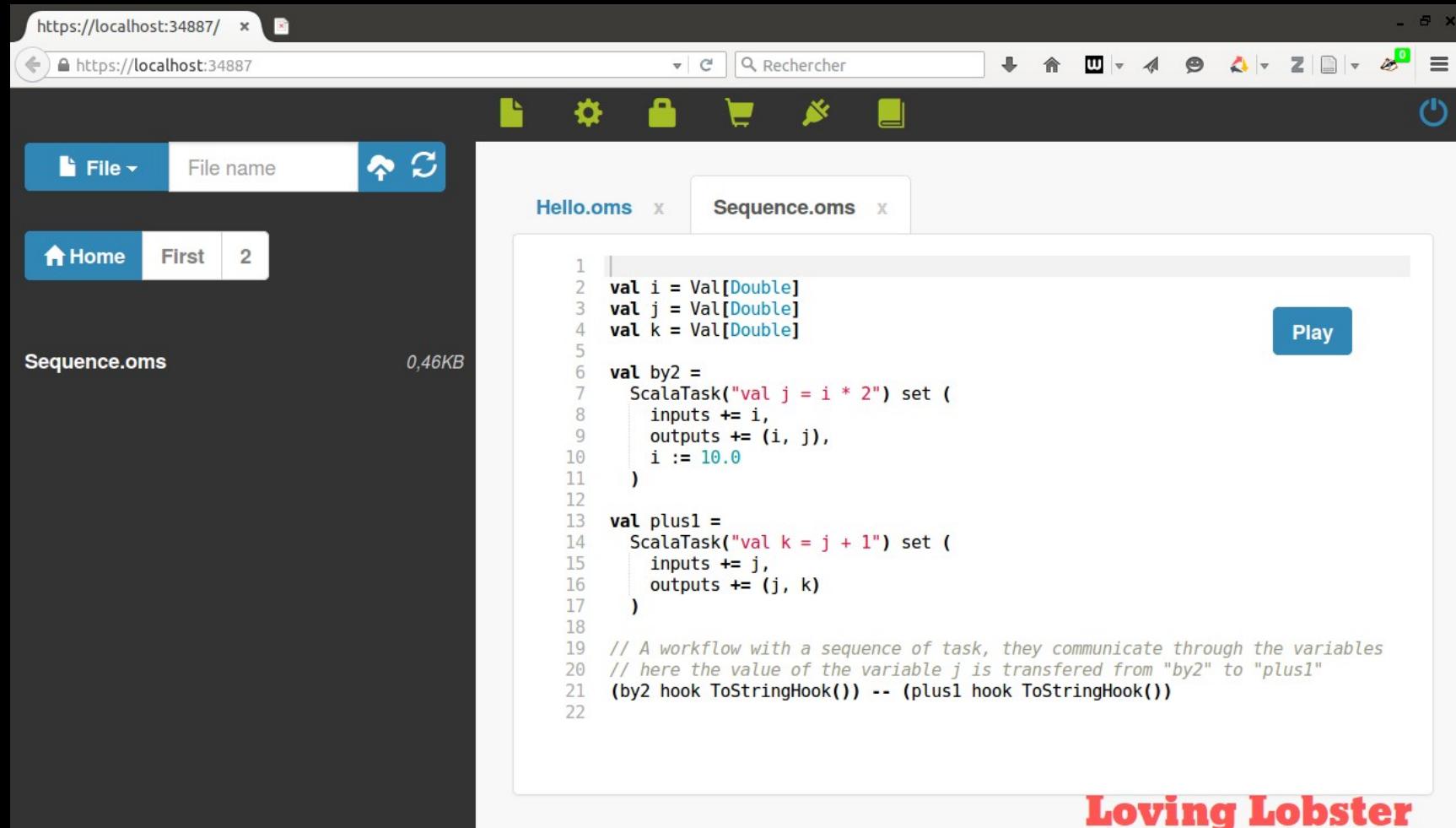
Running...

# First workflow



Display the result.

# The sequence



A screenshot of a web browser window displaying a Scala code editor. The URL is `https://localhost:34887/`. The code editor shows two tabs: `Hello.oms` and `Sequence.oms`. The `Sequence.oms` tab is active, containing the following Scala code:

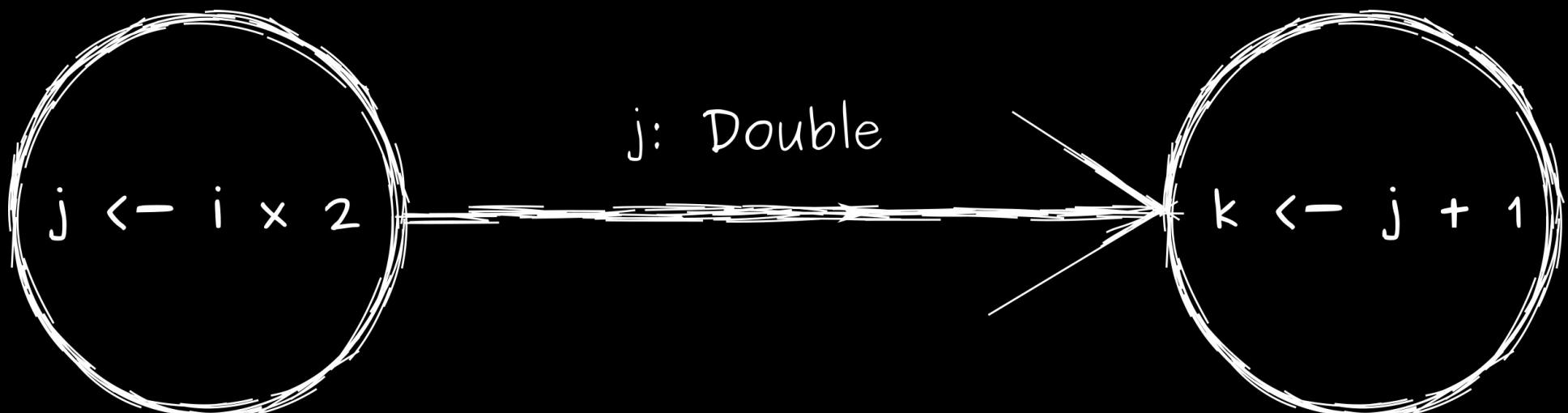
```
1  |||  
2  val i = Val[Double]  
3  val j = Val[Double]  
4  val k = Val[Double]  
5  
6  val by2 =  
7    ScalaTask("val j = i * 2") set (  
8      inputs += i,  
9      outputs += (i, j),  
10     i := 10.0  
11   )  
12  
13  val plus1 =  
14    ScalaTask("val k = j + 1") set (  
15      inputs += j,  
16      outputs += (j, k)  
17    )  
18  
19 // A workflow with a sequence of task, they communicate through the variables  
20 // here the value of the variable j is transferred from "by2" to "plus1"  
21 (by2 hook ToStringHook()) -- (plus1 hook ToStringHook())  
22
```

On the right side of the code editor, there is a blue **Play** button. At the bottom right of the entire window, the text **Loving Lobster** is displayed in red.

Go to the directory "2"  
Click on "Sequence.oms"

# The sequence

Tasks communicate through variables.

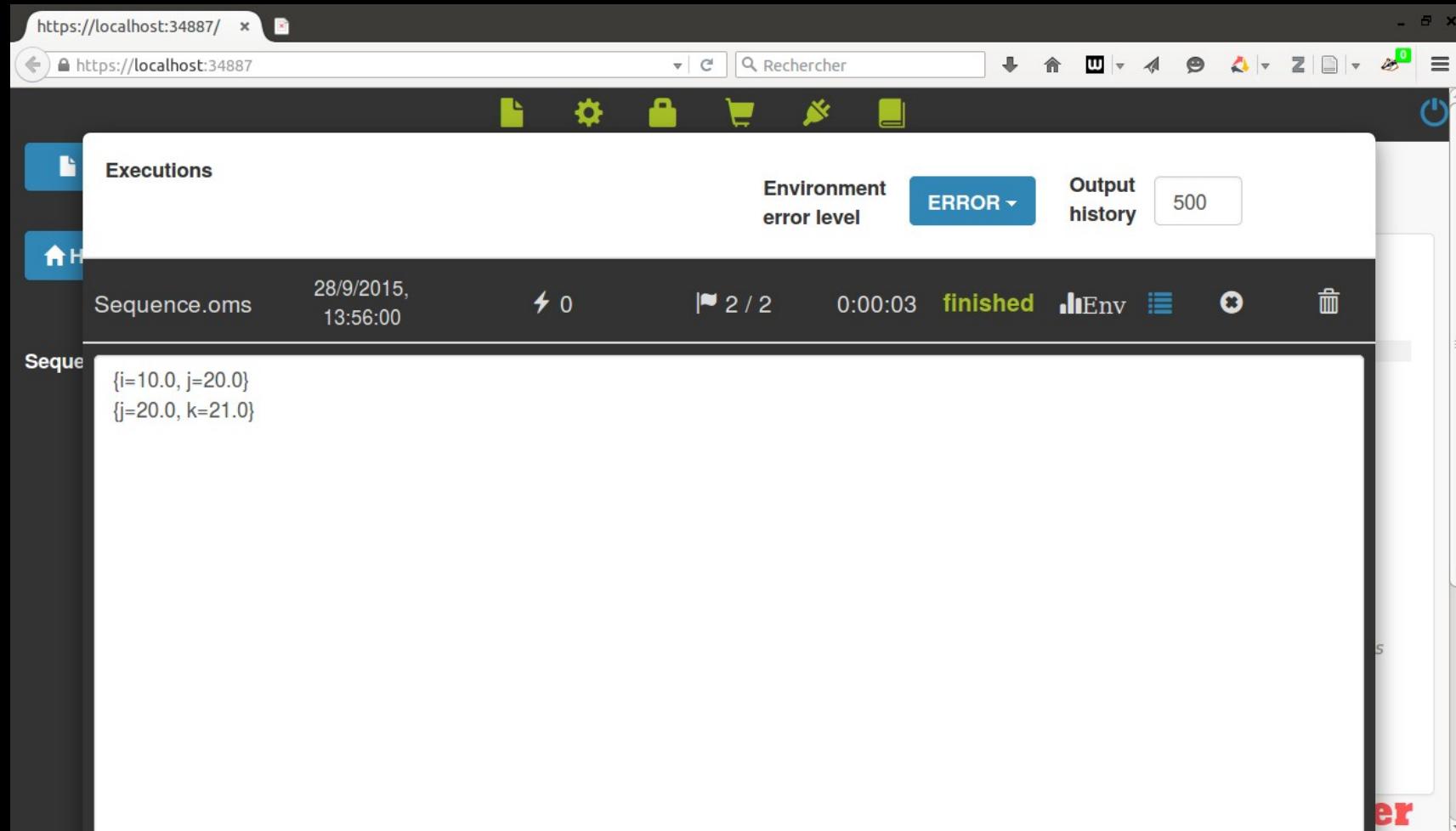


# The sequence

A sequence of task is represented by --

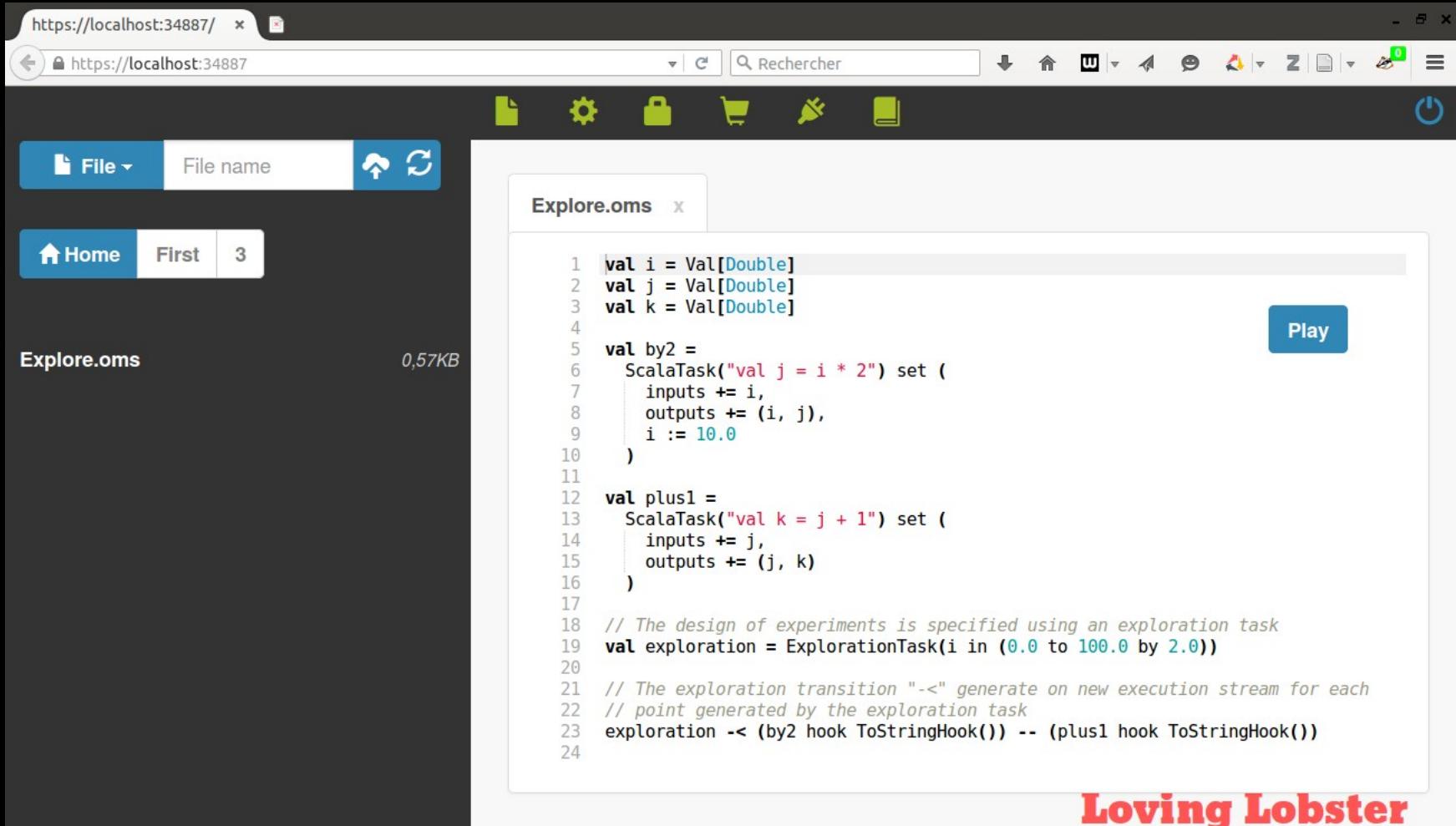
```
18
19 // A workflow with a sequence of task, they communicate through the variables
20 // here the value of the variable j is transferred from "by2" to "plus1"
21 (by2 hook ToStringHook()) -- (plus1 hook ToStringHook())
22
```

# The sequence



Run the workflow.

# The exploration



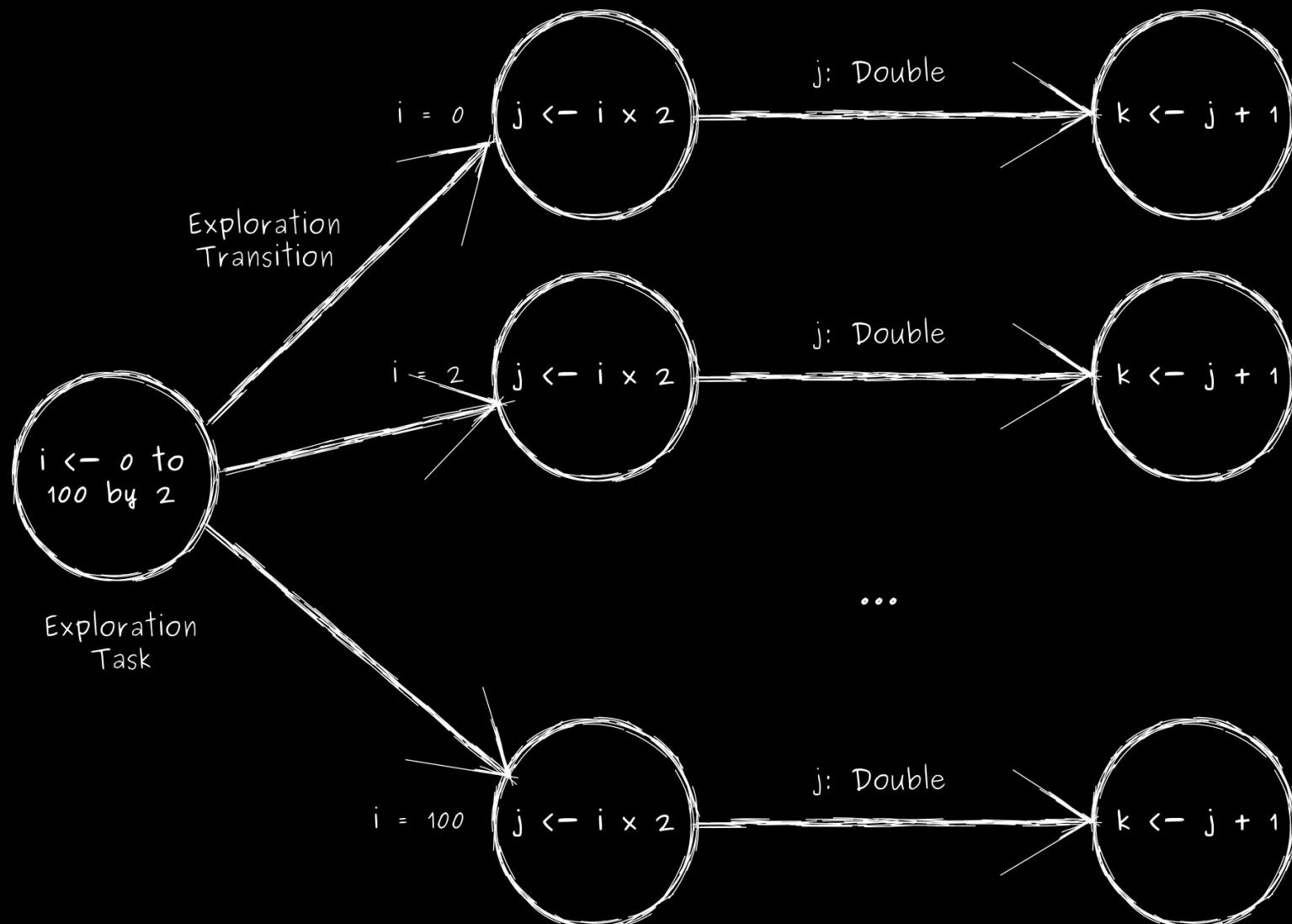
A screenshot of a web browser window displaying a file named "Explore.oms". The browser interface includes a header with a search bar and various icons, a toolbar with file operations like "File", "Upload", and "Refresh", and a navigation bar with "Home", "First", and "3". The main content area shows the code for "Explore.oms" with line numbers 1 through 24. The code defines Scala tasks for generating values i, j, and k, and for performing operations like doubling and adding 1. It also specifies an exploration task for value i ranging from 0.0 to 100.0. A "Play" button is visible on the right side of the code editor.

```
1 val i = Val[Double]
2 val j = Val[Double]
3 val k = Val[Double]
4
5 val by2 =
6   ScalaTask("val j = i * 2") set (
7     inputs += i,
8     outputs += (i, j),
9     i := 10.0
10   )
11
12 val plus1 =
13   ScalaTask("val k = j + 1") set (
14     inputs += j,
15     outputs += (j, k)
16   )
17
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))
20
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())
```

**Loving Lobster**

Go to the directory "3"  
Click on "Sequence.oms"

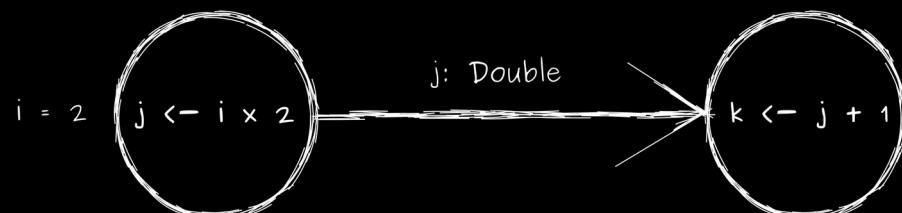
# The exploration



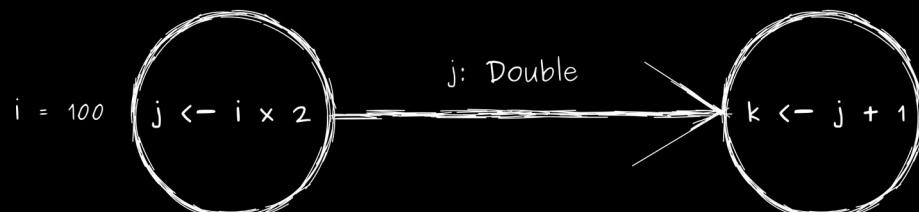
Exploration Task + Exploration Transition

# 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))  
20
```

The exploration task contains the design of experiments.

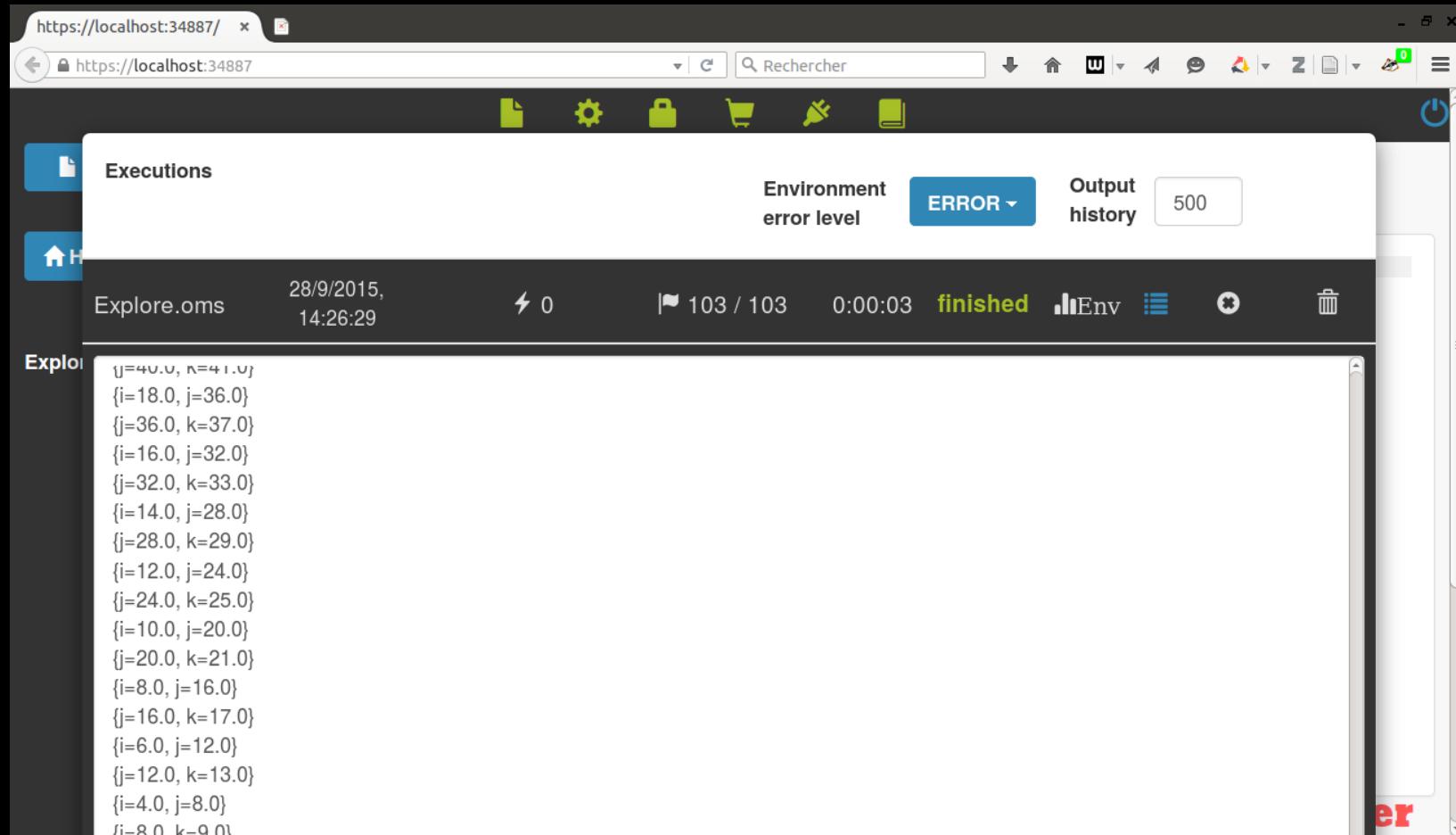
A design of experiments can compose many things (see the Sampling doc).

# 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

The screenshot shows a web-based interface for managing OpenMOLE projects. At the top, a browser window displays the URL <https://localhost:34887/>. Below the browser is a toolbar with various icons. On the left, there's a sidebar with a 'File' dropdown, a 'File name' input field, and upload/download/cancel buttons. The main area shows a file named 'Store.oms' with a size of 0,64KB. The file content is a Scala script:

```
1 val i = Val[Double]
2 val j = Val[Double]
3 val k = Val[Double]
4
5 val by2 =
6   ScalaTask("val j = i * 2") set (
7     inputs += i,
8     outputs += (i, j),
9     i := 10.0
10   )
11
12 val plus1 =
13   ScalaTask("val k = j + 1") set (
14     inputs += j,
15     outputs += (j, k)
16   )
17
18 val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))
19
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)
24
```

A 'Play' button is located in the bottom right corner of the code editor. The bottom of the page features a red banner with the text 'Loving Lobster'.

<https://localhost:34887/downloadFile?path=projects/First/4/Store.oms>

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)  
24
```

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

A screenshot of a web browser displaying the OpenMOLE interface at <https://localhost:34887/>. The interface includes a top navigation bar with icons for file operations, settings, and help. Below the bar, there's a file list on the left showing a file named "results" (1.07KB) and another file named "Store.oms" (0.64KB). On the right, a code editor window titled "Store.oms" contains Scala code for a task named "by2". A red arrow points from the text "Refresh" to the refresh icon in the top right corner of the code editor window. A blue "Play" button is located to the right of the code editor. At the bottom right of the slide, the text "Loving Lobster" is displayed in red.

```
val by2 = ScalaTask("val i = j * 2") set (
    inputs += i,
    outputs += (i, j),
    i := 10.0
)

val plus1 =
ScalaTask("val k = j + 1") set (
    inputs += j,
    outputs += (j, k)
)

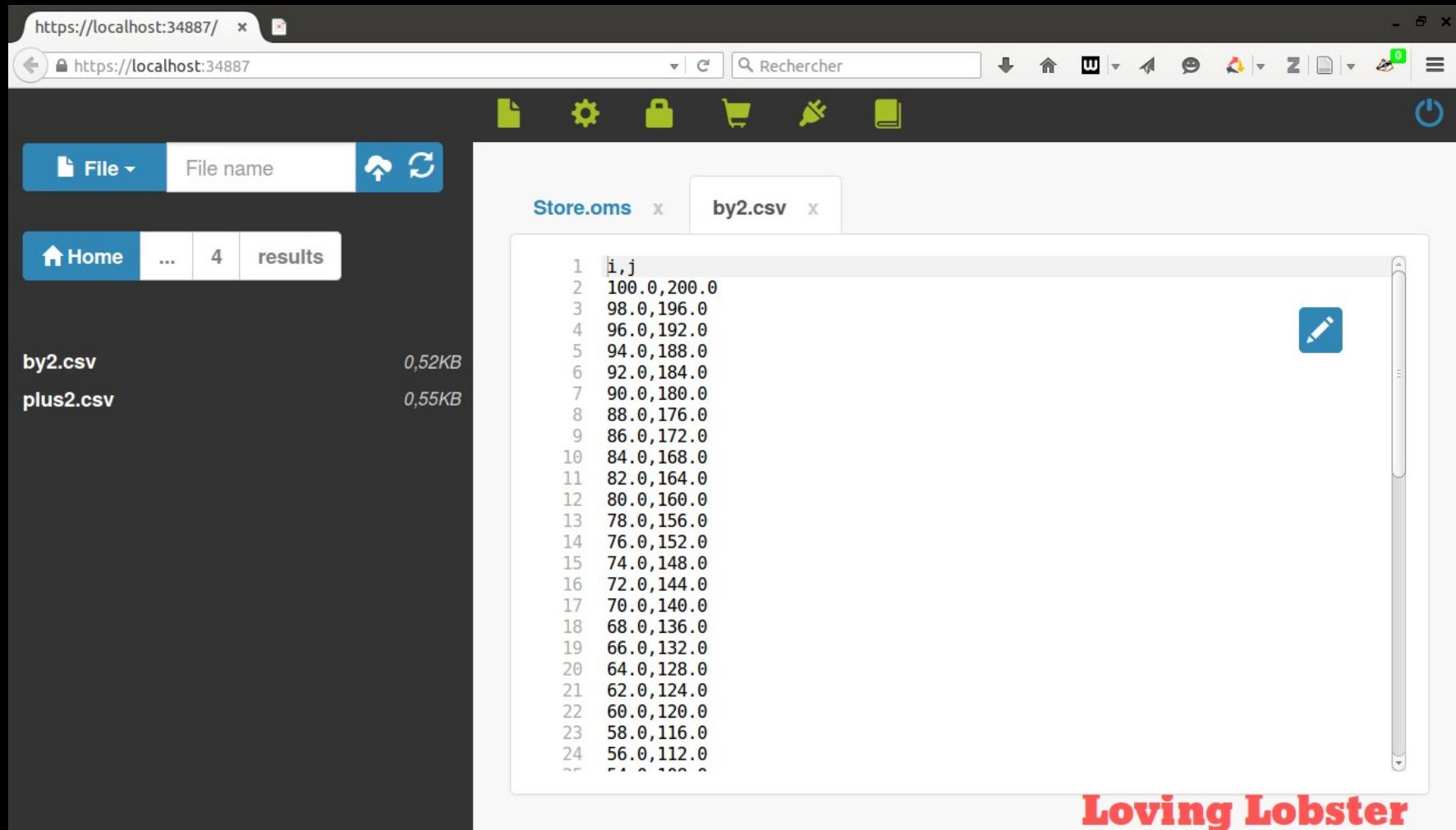
val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))

// In OpenMOLE there are different kinds of hooks. This one has been designed
// to store variables into a CSV file.
val by2Hook = AppendToCSVFileHook(workDirectory / "results/by2.csv", i, j)
val plus1Hook = AppendToCSVFileHook(workDirectory / "results/plus2.csv", j, k)
exploration -< (by2 hook by2Hook) -- (plus1 hook plus1Hook)
```

Loving Lobster

Run it and refresh.

# Store results in files



The screenshot shows a web browser window with the URL <https://localhost:34887/>. The interface is a file manager or storage service. On the left, there's a sidebar with a 'File' dropdown, a 'File name' search bar, and a toolbar with icons for file operations. Below the sidebar, there are links for 'Home', '...', '4 results', 'by2.csv' (0,52KB), and 'plus2.csv' (0,55KB). The main area displays two files: 'Store.oms' and 'by2.csv'. The 'by2.csv' file is open, showing its contents in a table-like format. The data consists of 24 rows, each containing two columns: 'i,j' and a numerical value. A blue edit icon is located in the top right corner of the 'by2.csv' preview area. At the bottom right of the main content area, the text 'Loving Lobster' is displayed in red.

i,j	
2	100.0,200.0
3	98.0,196.0
4	96.0,192.0
5	94.0,188.0
6	92.0,184.0
7	90.0,180.0
8	88.0,176.0
9	86.0,172.0
10	84.0,168.0
11	82.0,164.0
12	80.0,160.0
13	78.0,156.0
14	76.0,152.0
15	74.0,148.0
16	72.0,144.0
17	70.0,140.0
18	68.0,136.0
19	66.0,132.0
20	64.0,128.0
21	62.0,124.0
22	60.0,120.0
23	58.0,116.0
24	56.0,112.0

Results have been stored in the "results" folder.

# Store results in files

A screenshot of a web browser window titled "https://localhost:34887/". The main content area shows a file named "by2.csv" with the following data:

i,j
100.0,200.0
98.0,196.0
96.0,192.0
94.0,188.0
92.0,184.0
90.0,180.0
88.0,176.0
86.0,172.0
84.0,168.0
82.0,164.0
80.0,160.0
78.0,156.0
76.0,152.0
74.0,148.0
72.0,144.0
70.0,140.0
68.0,136.0
66.0,132.0
64.0,128.0
62.0,124.0
60.0,120.0
58.0,116.0
56.0,112.0

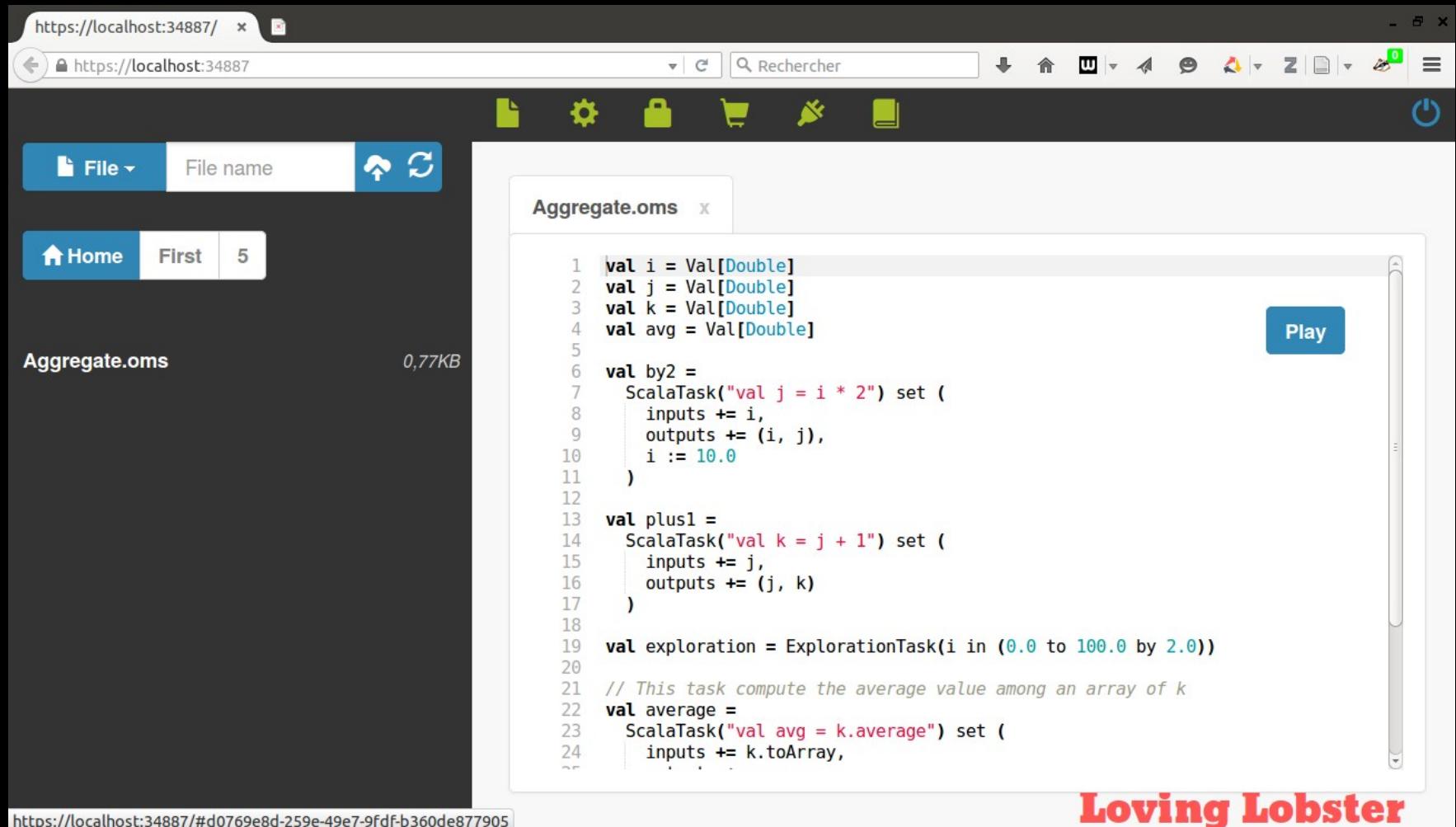
The browser interface includes a "File" menu, a search bar, and various icons. A red arrow points from the "Download" button next to the file listing on the left towards the file content area, with the word "Download" written in red over the arrow.

**Loving Lobster**

<https://localhost:34887/downloadFile?path=projects/First/4/results/by2.csv>

You can download it.

# Aggregate



The screenshot shows a web browser window with the URL <https://localhost:34887/>. The page displays a file named "Aggregate.oms" with a size of 0,77KB. The file content is a Scala code snippet:

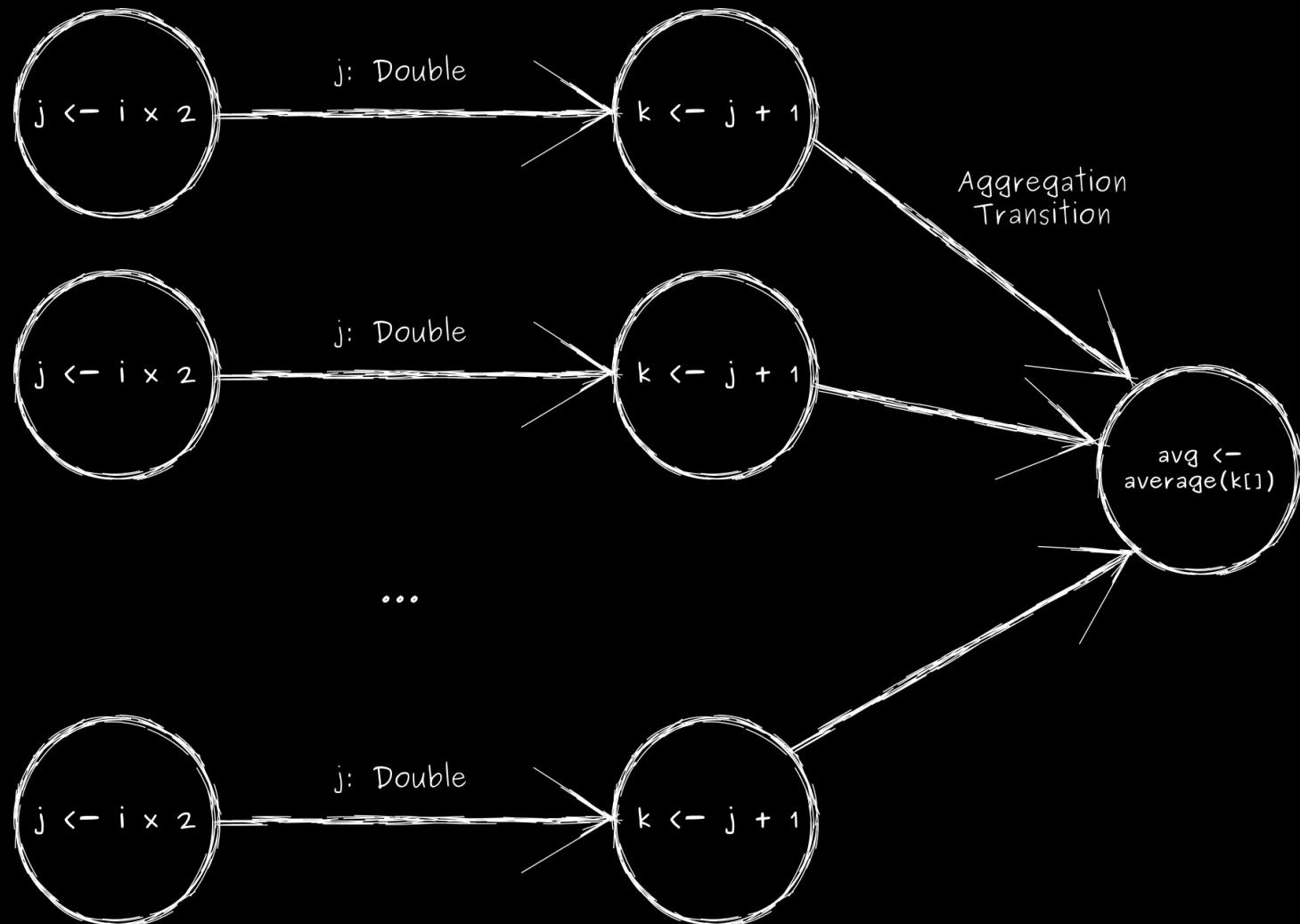
```
1 val i = Val[Double]
2 val j = Val[Double]
3 val k = Val[Double]
4 val avg = Val[Double]
5
6 val by2 =
7   ScalaTask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13 val plus1 =
14   ScalaTask("val k = j + 1") set (
15     inputs += j,
16     outputs += (j, k)
17   )
18
19 val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))
20
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,
```

A "Play" button is visible on the right side of the code editor. The browser interface includes a toolbar with various icons and a navigation bar with "Home", "First", and "5". The address bar shows the URL again.

Loving Lobster

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   )
27
```

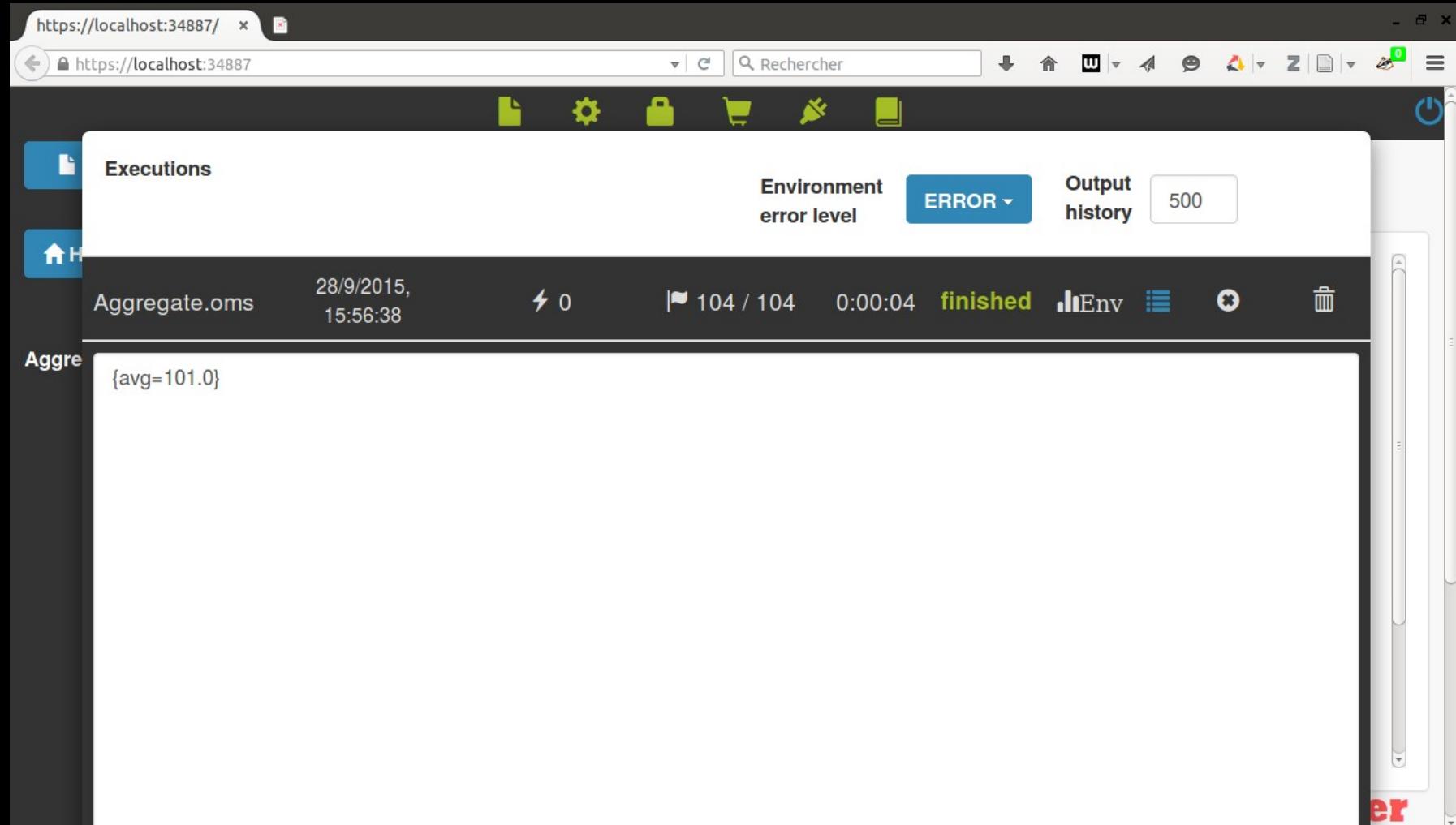
Computes the average over the values of k.  
Takes the input "k: Array[Double]".  
Produces "avg: Double".

# 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())
```

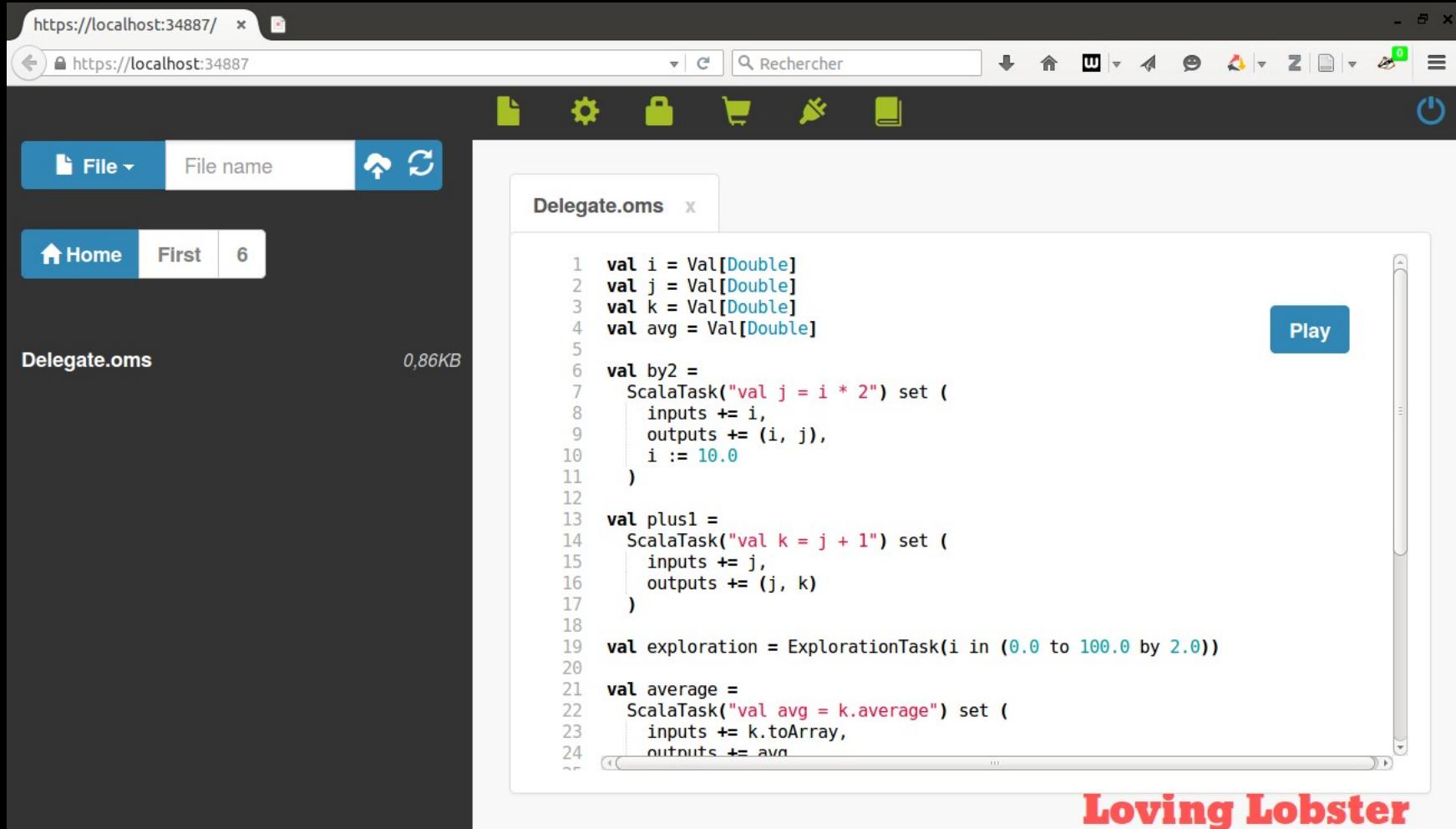
The aggregation transition is noted ">-". It gathers the results among the execution streams produced by the matching "-<".

# Aggregate



Run it.

# Delegate the execution



The screenshot shows a web browser window with the URL <https://localhost:34887/>. The page displays a file named "Delegate.oms". The file content is as follows:

```
1 val i = Val[Double]
2 val j = Val[Double]
3 val k = Val[Double]
4 val avg = Val[Double]
5
6 val by2 =
7   ScalaTask("val j = i * 2") set (
8     inputs += i,
9     outputs += (i, j),
10    i := 10.0
11  )
12
13 val plusl =
14   ScalaTask("val k = j + 1") set (
15     inputs += j,
16     outputs += (j, k)
17   )
18
19 val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))
20
21 val average =
22   ScalaTask("val avg = k.average") set (
23     inputs += k.toArray,
24     outputs += avg
25 )
```

A "Play" button is visible on the right side of the code editor.

Loving Lobster

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 delegate tasks to execution environments  
36 exploration -< (by2 on env) -- (plus1 on env) >- (average hook ToStringHook())
```

Use the “on” keyword to delegate the task executions.

# Delegate the execution

A screenshot of a web browser window displaying a user interface for managing executions. The URL is <https://localhost:34887/>. The interface has a dark theme with green and blue highlights. At the top, there are various icons and a search bar labeled "Rechercher". Below the header, there's a toolbar with icons for file operations, settings, and navigation. A main panel titled "Executions" shows a single entry: "Delegate.oms" from "29/9/2015, 13:34:22". The status bar indicates "finished" with a duration of "0:00:06". A red arrow points from the text "Display execution environments" to the "details" link next to the execution entry. The bottom half of the screen shows a code editor with the following Scala code:

```
13 val plus1 =
14   ScalaTask("val k = j + 1") set (
15     inputs += j,
16     outputs += (j, k)
17   )
18
19 val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))
20
21 val average =
22   ScalaTask("val avg = k.average") set (
23     inputs += k.toArray,
24     outputs += avg
25   )
26
27 // Run 4 parallel processes on the local computer
28 val env = LocalEnvironment(4)
29
```

**Loving Lobster**

Run it!

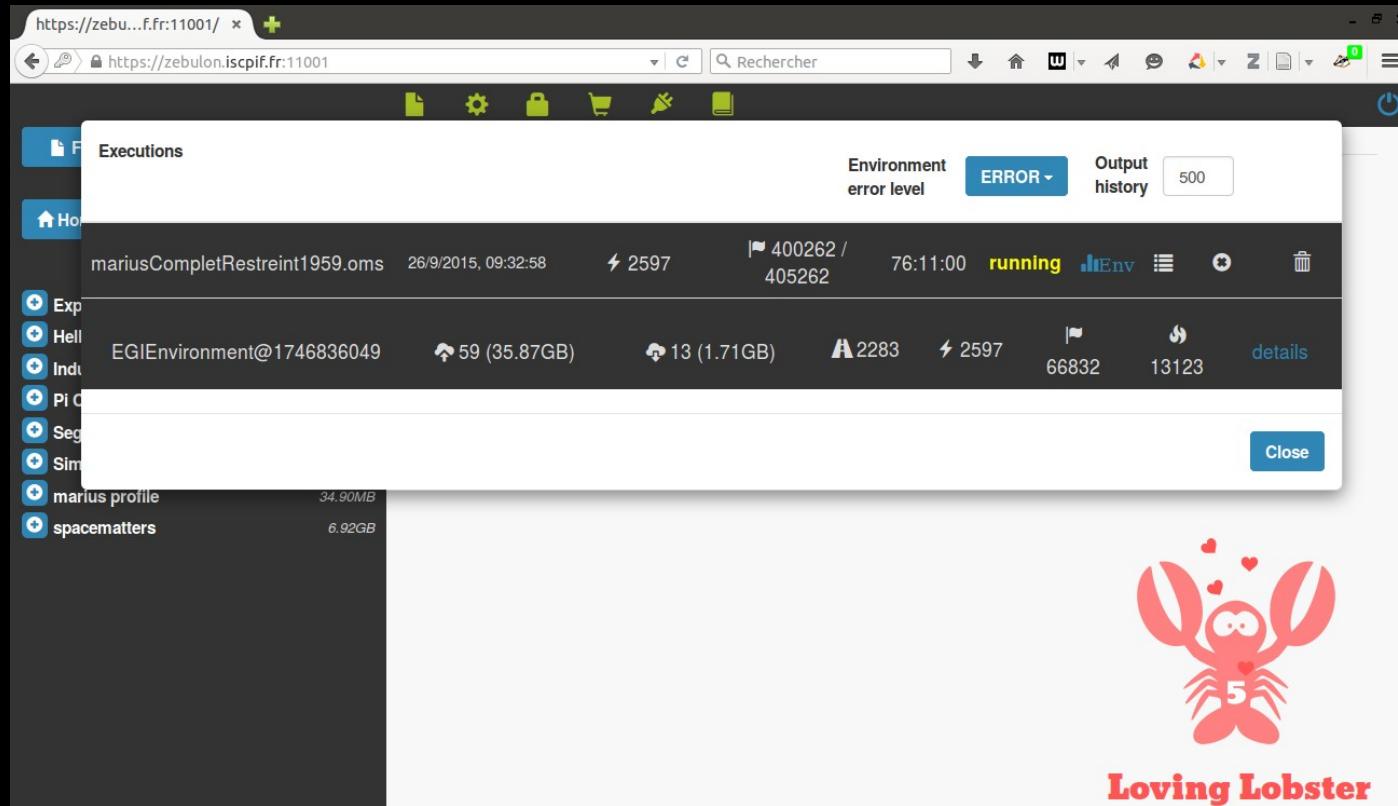
# Delegate the execution

Many execution environments are supported:

- most cluster systems
- the EGI Grid
- multi-core servers
- desktop-grid
- ... cloud is planned

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

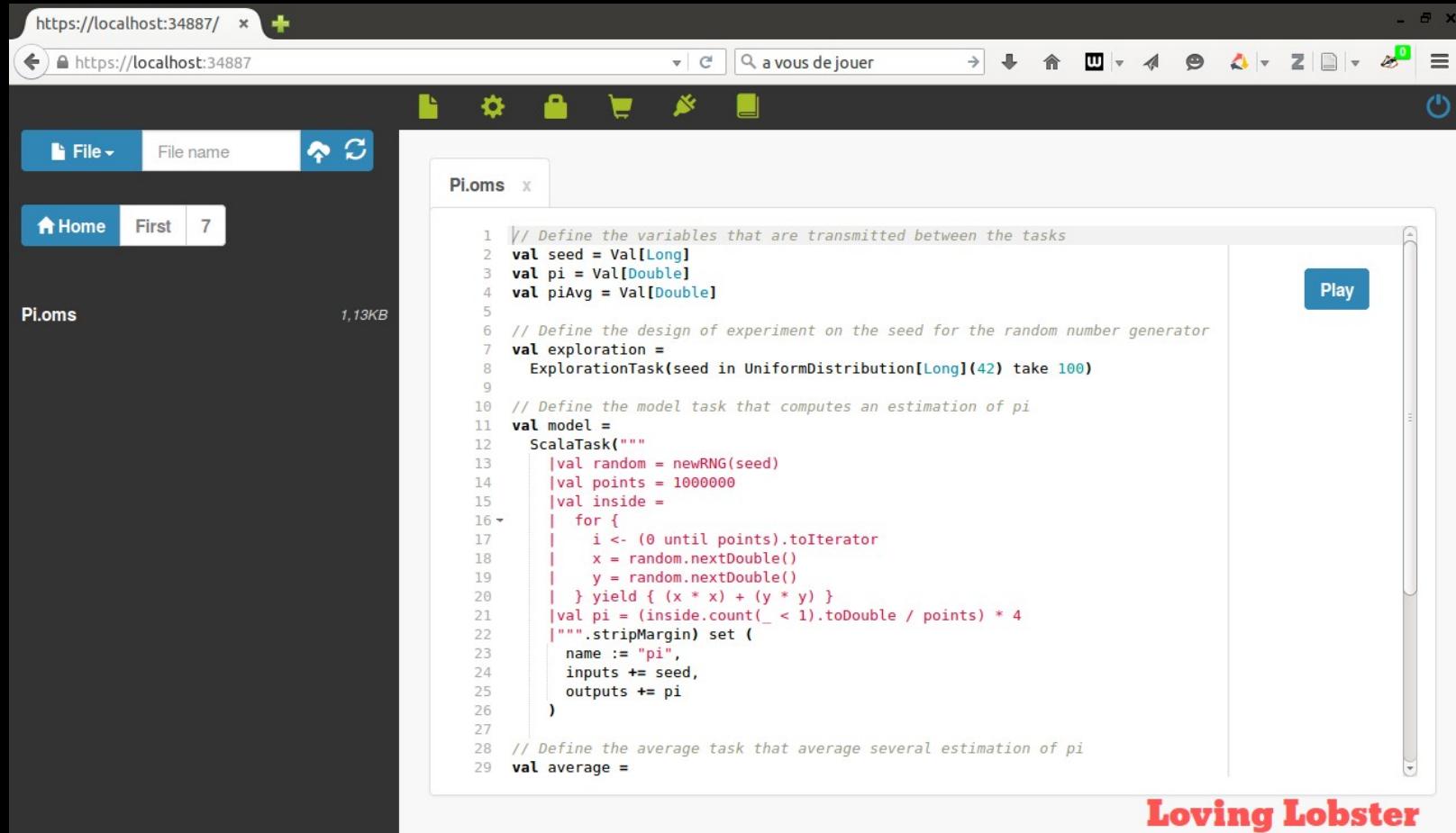
# Delegate the execution



OpenMOLE scales!

Millions of jobs, +100GB of data

# And now, play on!



The screenshot shows a web-based Scala code editor interface. The URL is <https://localhost:34887/>. The title bar says "Pi.oms". The code editor displays the following Scala code:

```
1 // Define the variables that are transmitted between the tasks
2 val seed = Val[Long]
3 val pi = Val[Double]
4 val piAvg = Val[Double]
5
6 // Define the design of experiment on the seed for the random number generator
7 val exploration =
8   ExplorationTask(seed in UniformDistribution[Long](42) take 100)
9
10 // Define the model task that computes an estimation of pi
11 val model =
12   ScalaTask("""
13     |val random = newRNG(seed)
14     |val points = 1000000
15     |val inside =
16     |  for {
17     |    i <- (0 until points).toIterator
18     |    x = random.nextDouble()
19     |    y = random.nextDouble()
20     |  } yield { (x * x) + (y * y) }
21     |val pi = (inside.count(_ < 1).toDouble / points) * 4
22     |""".stripMargin) set (
23       name := "pi",
24       inputs += seed,
25       outputs += pi
26     )
27
28 // Define the average task that average several estimation of pi
29 val average =
```

On the right side of the code editor, there is a "Play" button. At the bottom right of the window, the text "Loving Lobster" is displayed.

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.