

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/jayatanaag.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 $
```

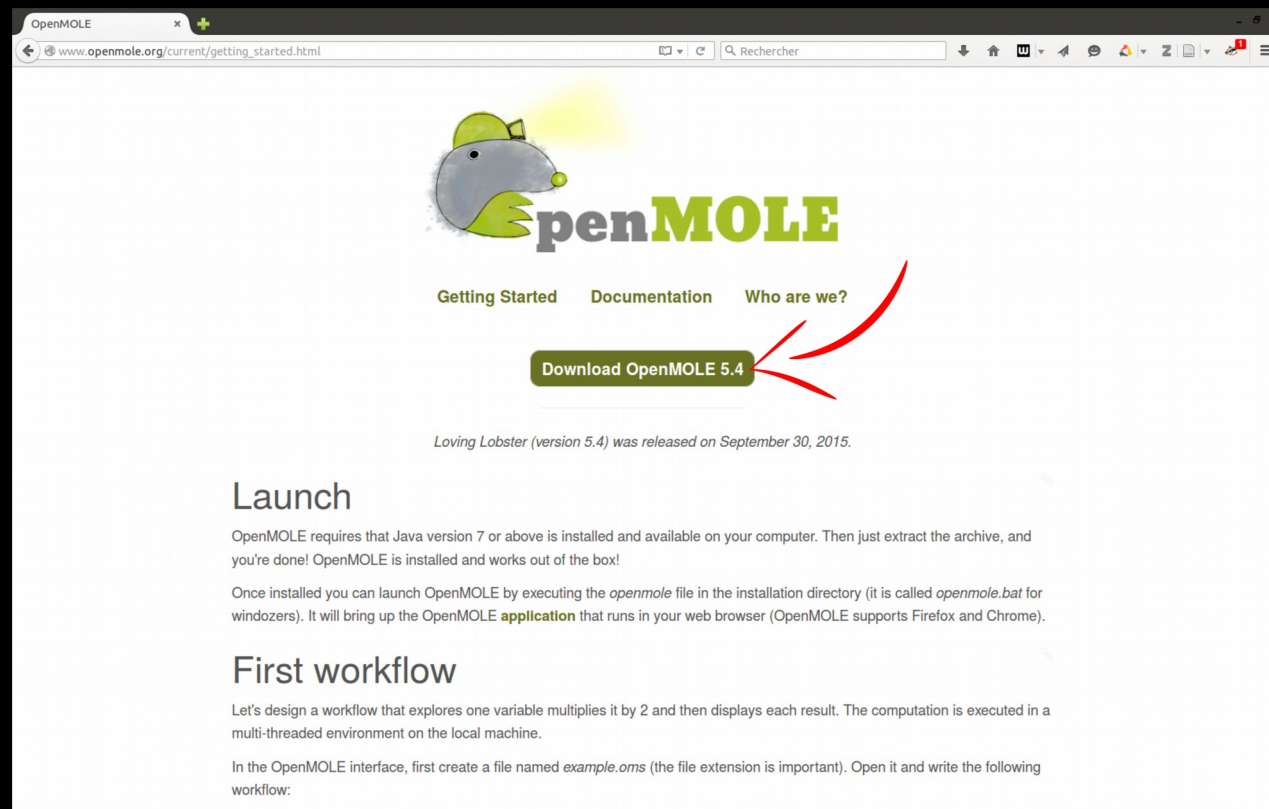
# Prerequisite

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

# Download

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

Go to: Getting started.

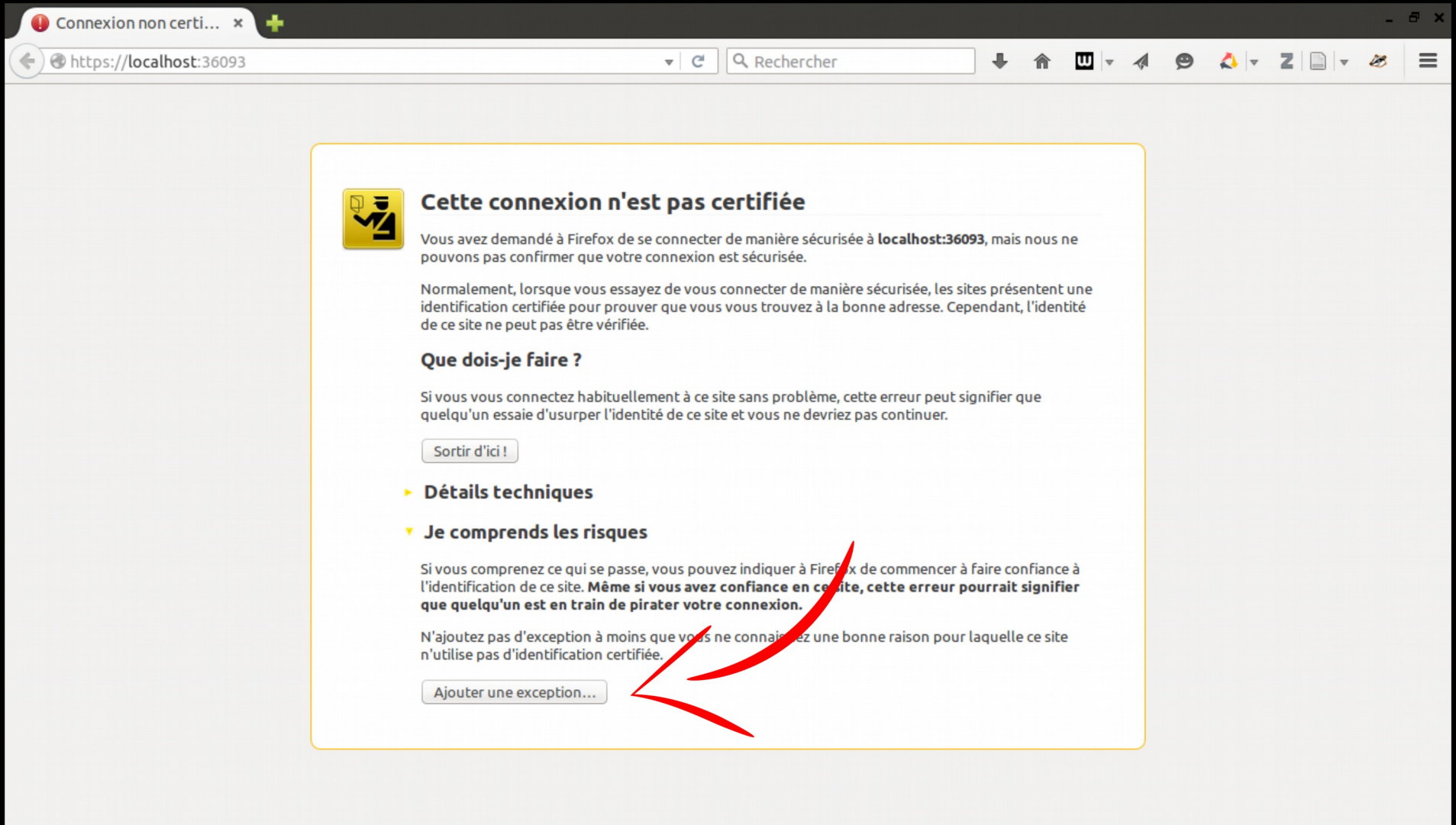


# Launch

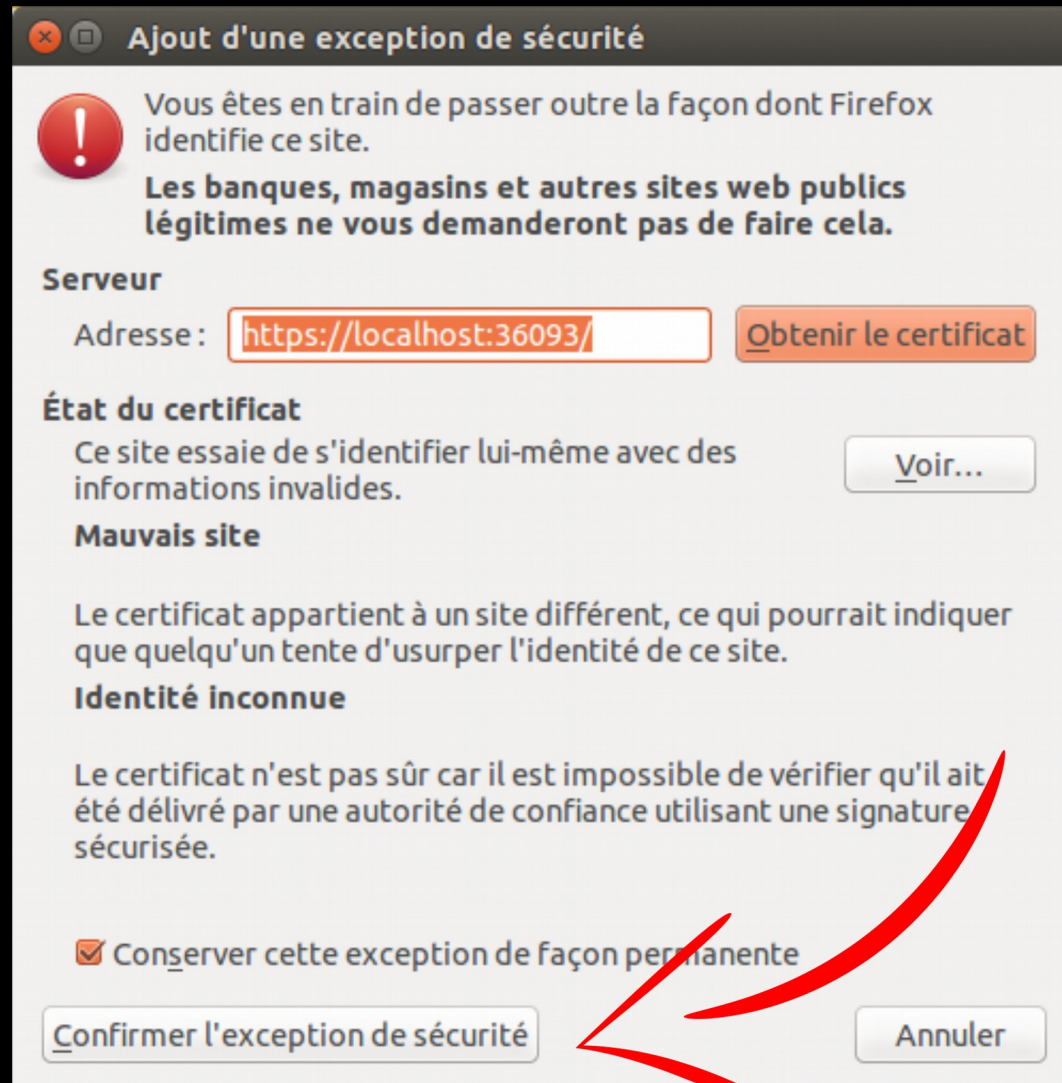
Run:

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

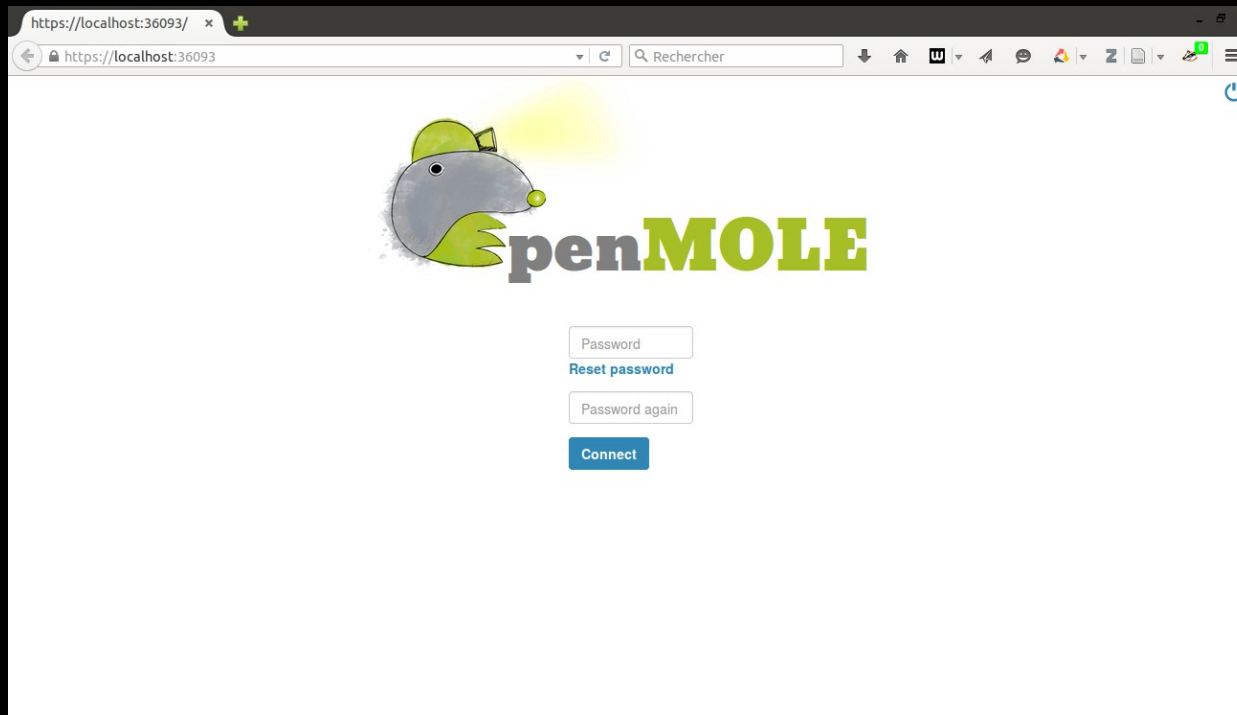
# Insecure connection



# Insecure connection



# OpenMOLE password



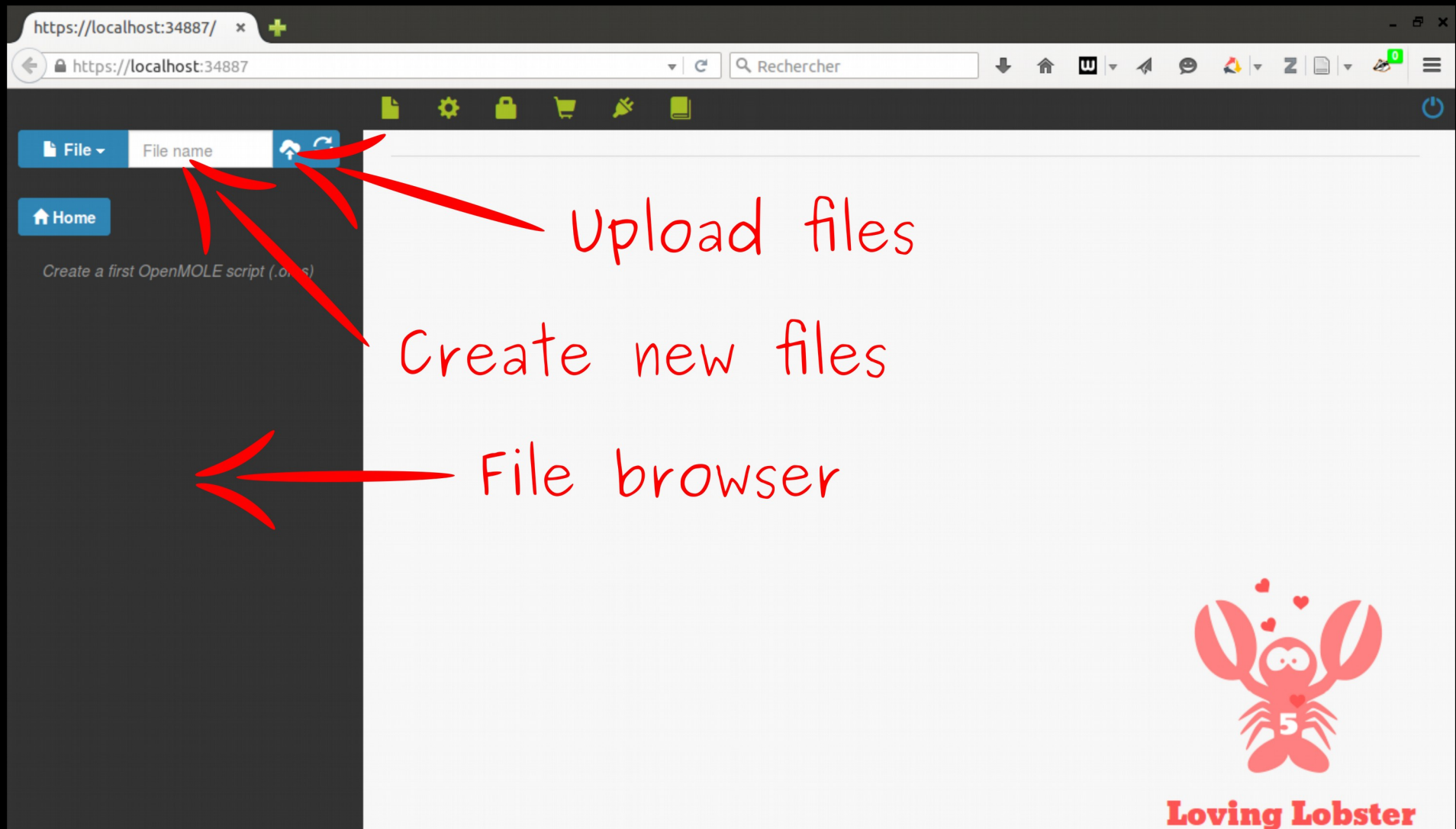
The screenshot shows a web browser window with the address bar displaying `https://localhost:36093/`. The page features the OpenMOLE logo, which consists of a cartoon mole character wearing a green cap and holding a small yellow object, followed by the text "openMOLE" in a green, stylized font. Below the logo, there is a form for password creation with the following elements:

- A text input field labeled "Password".
- A blue link labeled "Reset password".
- A text input field labeled "Password again".
- A blue button labeled "Connect".

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



# Welcome to OpenMOLE

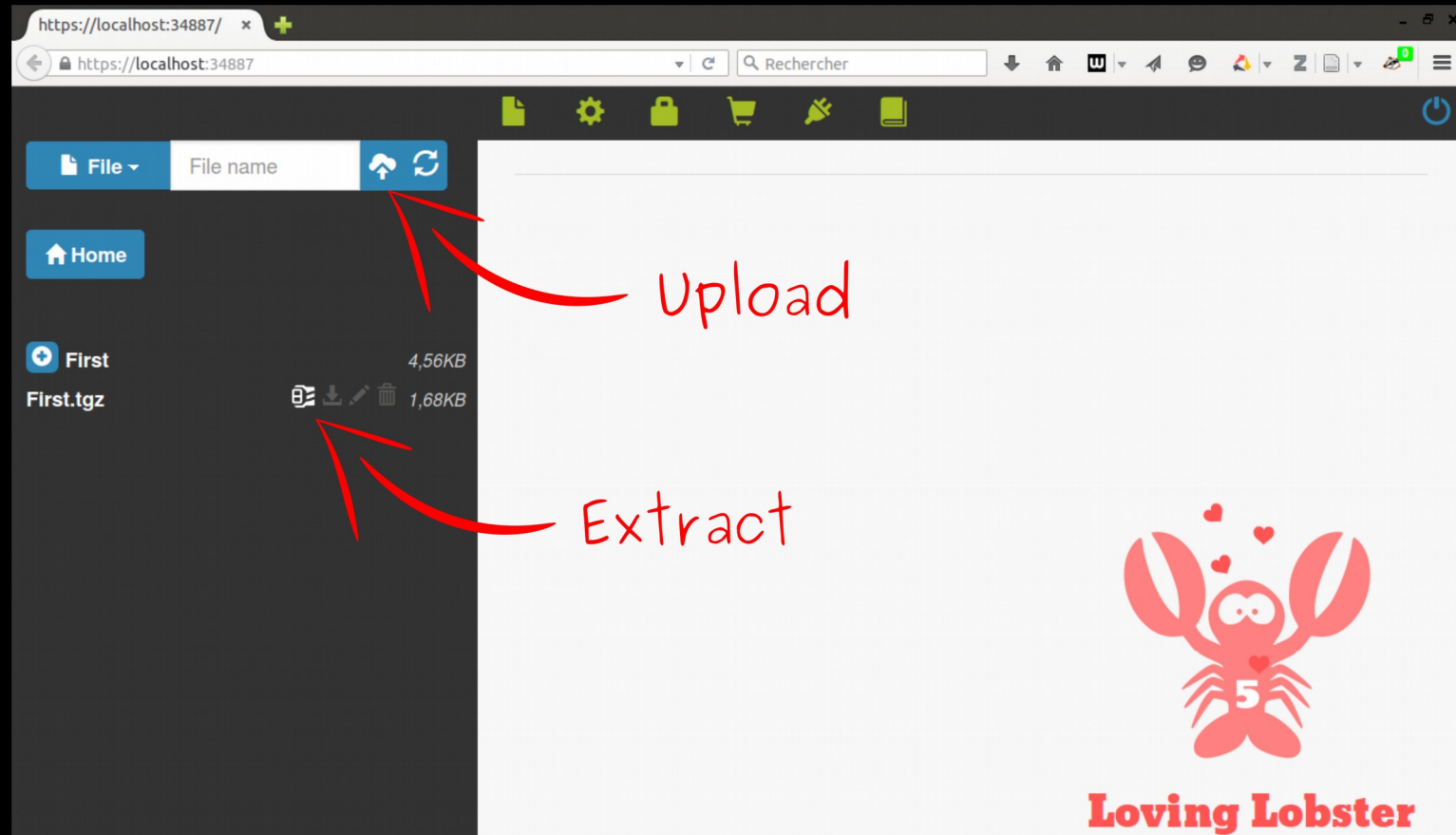


# First workflow

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

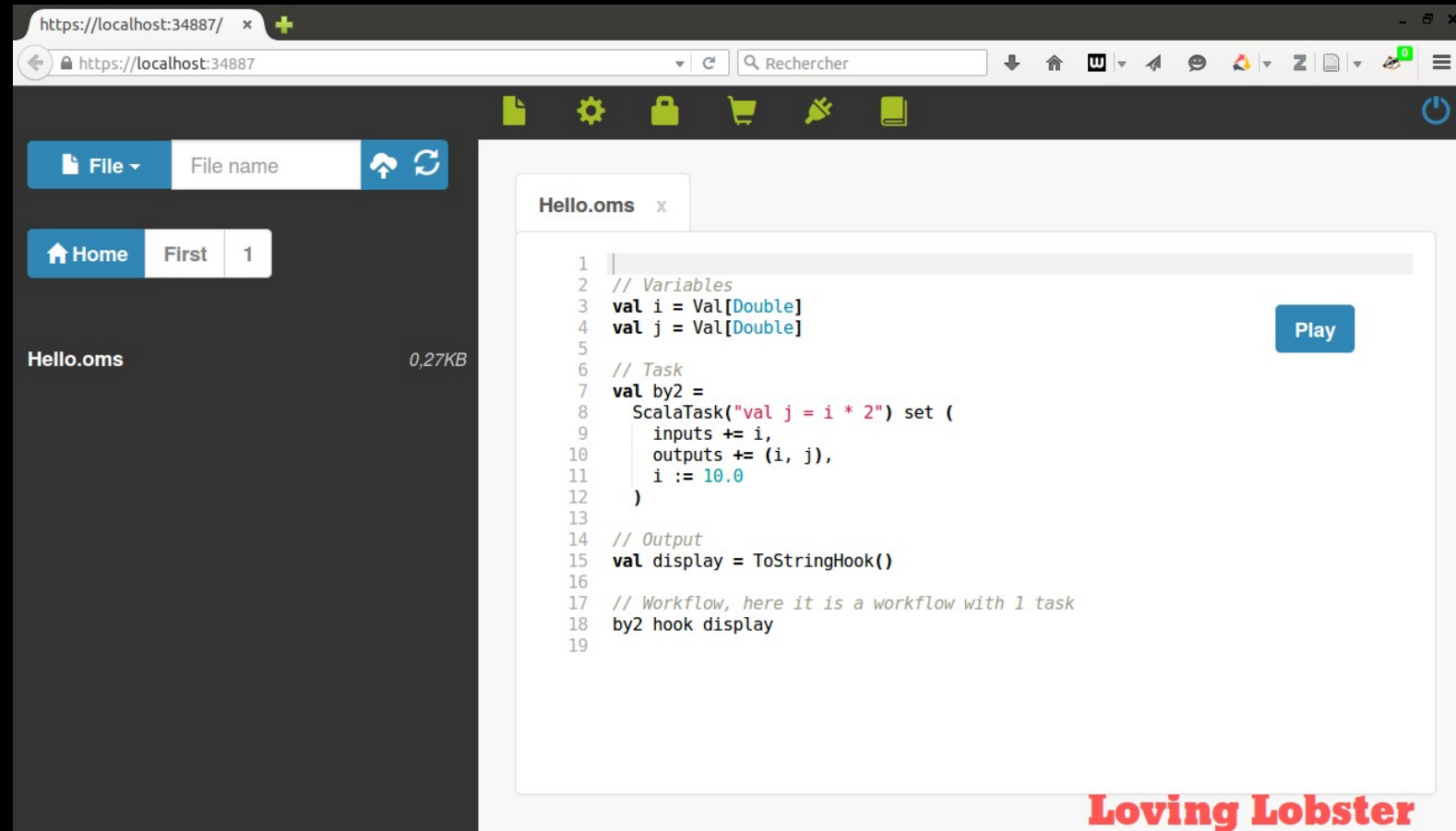
Download First.tgz

# First workflow



Upload the archive  
Extract it

# First workflow



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

Loving Lobster

Click on the folder "First"  
Click on the folder "1"  
Click on the file "Hello.oms"

# First workflow

The screenshot shows a web browser at <https://localhost:34887/> displaying a workflow editor. The interface includes a sidebar with a 'File' menu and a 'Home' button. The main area shows a file named 'Hello.oms' with a size of 0,27KB. The code is as follows:

```
1 // Variables
2
3 val i = Val[Double]
4 val j = Val[Double]
5
6 // Task
7 by2 =
8   ScalaTask("val j = i * 2") set (
9     inputs += i,
10    outputs += (i, j),
11    i := 10.0
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 red circle highlights the variable declarations on lines 3 and 4. A 'Play' button is visible on the right side of the code editor.

**Loving Lobster**

Express the dataflow  
Variables are typed

# First workflow

The screenshot shows a web-based IDE interface for a workflow named "Hello.oms". The interface includes a file explorer on the left with a "File" button and a "File name" input field. Below the file explorer is a breadcrumb trail: "Home > First > 1". The main editor area displays the following Scala code:

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

A red circle highlights the task definition (lines 7-12). A "Play" button is located to the right of the code editor. The file "Hello.oms" is listed in the file explorer with a size of 0,27KB. The bottom right corner of the interface features the text "Loving Lobster".

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

# First workflow

The screenshot shows the OMS web interface at <https://localhost:34887/>. The left sidebar contains a file explorer with a 'File' button and a search bar. Below it, there are tabs for 'Home', 'First', and '1'. The main area displays the 'Hello.oms' file, which is 0,27KB in size. The code editor shows the following Scala code:

```
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 circle highlights the `ToStringHook()` function in the output section. A 'Play' button is visible on the right side of the code editor. The bottom right corner of the interface features the text 'Loving Lobster'.

Express output operations  
Always runs locally



# First workflow

The screenshot shows a web application running on `https://localhost:34887/`. The interface includes a sidebar with a 'File' menu and tabs for 'Home', 'First', and '1'. The main area displays a code editor for a file named 'Hello.oms' (0.27KB). The code is written in Scala and defines a workflow with one task and one hook. A red circle highlights the hook definition on line 18.

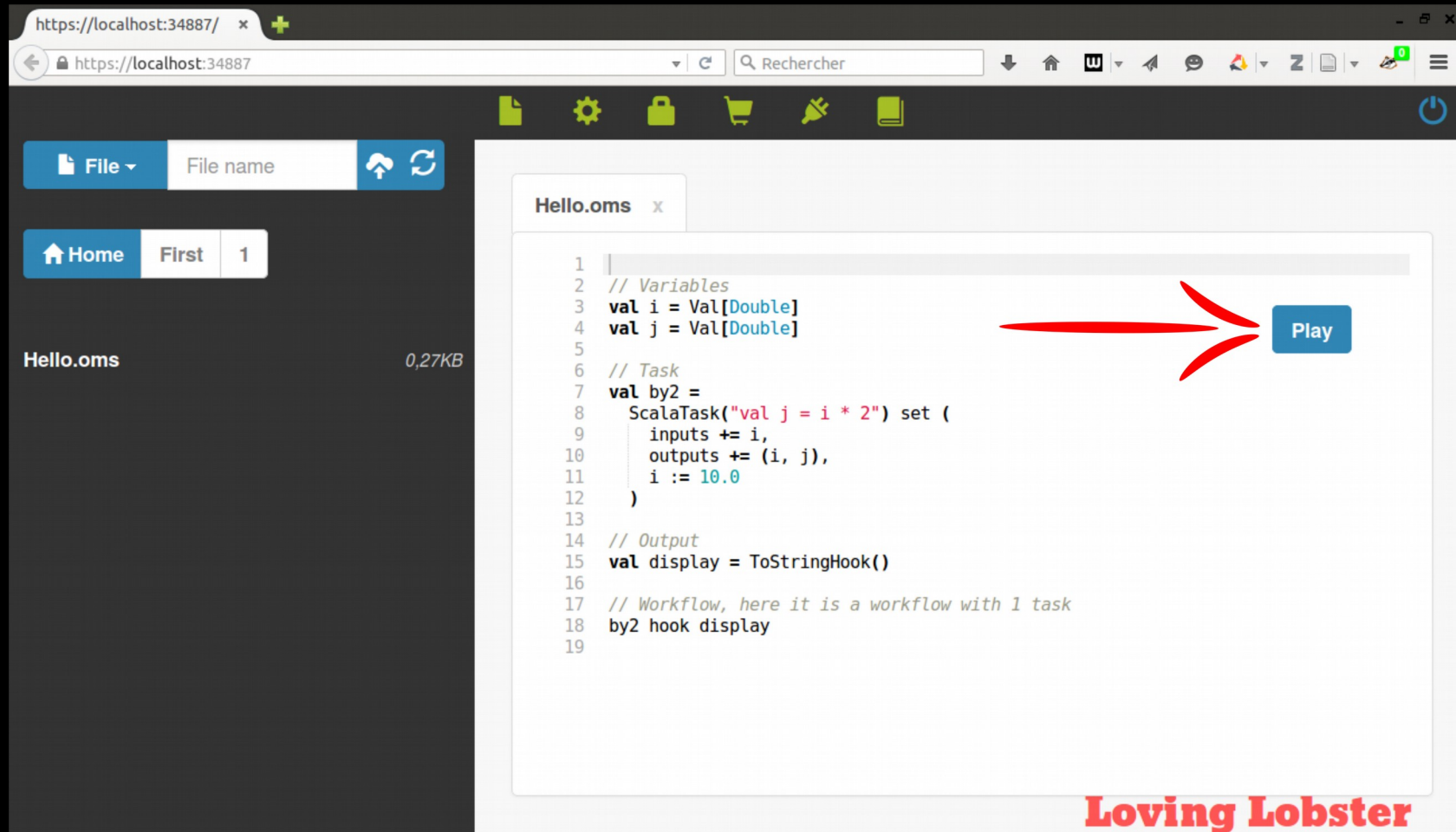
```
1  
2 // Variables  
3 val i = Val[Double]  
4 val j = Val[Double]  
5  
6 // Task  
7 val by2 =  
8   ScalaTask("val j = i * 2") set (  
9     inputs += i,  
10    outputs += (i, j),  
11    i := 10.0  
12  )  
13  
14 // Output  
15 val display = ToStringHook()  
16  
17 // Workflow, here it is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

A 'Play' button is located to the right of the code editor. The text 'Loving Lobster' is visible in the bottom right corner of the interface.

Express workflow composition  
Here: One task + One hook



# First workflow



The screenshot shows a web browser at `https://localhost:34887/` displaying the Hello.oms interface. On the left, there's a sidebar with a 'File' button, a 'File name' input, and a 'Home' button. Below this, a breadcrumb shows 'Home' > 'First' > '1'. The main area displays a file named 'Hello.oms' (0,27KB). The code editor shows the following Scala code:

```
1 |  
2 // Variables  
3 val i = Val[Double]  
4 val j = Val[Double]  
5  
6 // Task  
7 val by2 =  
8   ScalaTask("val j = i * 2") set (  
9     inputs += i,  
10    outputs += (i, j),  
11    i := 10.0  
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 red arrow points from the code editor to a blue 'Play' button on the right. The text 'Loving Lobster' is visible in the bottom right corner of the interface.

And now lets run!

# First workflow

The screenshot shows a web browser at `https://localhost:34887/` displaying a workflow execution interface. A modal window titled "Executions" is open, showing a table with one row for a job named "Hello.oms" started on "28/9/2015, 12:15:33". The job status is "running" (indicated by a yellow lightning bolt icon). The "Environment error level" is set to "ERROR" and "Output history" is 500. Below the table, there is a code editor showing a Scala snippet:

```
10 outputs += (i, j),
11 i := 10.0
12 )
13
14 // Output
15 val display = ToStringHook()
16
17 // Workflow, here it is a workflow with 1 task
18 by2 hook display
19
```

Two red arrows point from handwritten notes to the job status and the code editor. The note "One job running" points to the lightning bolt icon. The note "No job finished yet" points to the code editor. A "Close" button is visible in the bottom right of the modal.

**One job running**

**No job finished yet**

**Loving Lobster**

Running...

# First workflow

The screenshot shows a web browser at `https://localhost:34887/` displaying a workflow execution interface. The main content area shows the execution of `Hello.oms` on `28/9/2015, 12:15:33`, which is `finished` after `0:00:01`. The output is `{i=10.0, j=20.0}`. A red arrow points from the handwritten text "Display the output" to the output field. The interface includes a top navigation bar with icons for file, settings, lock, shopping cart, and document. The left sidebar has buttons for "Executions" and "Home". The top right of the execution card shows "Environment error level" set to "ERROR" and "Output history" set to "500".

Executions

Environment error level ERROR Output history 500

Hello.oms 28/9/2015, 12:15:33 0 1 / 1 0:00:01 finished Env

Hello.oms {i=10.0, j=20.0}

Display the output

Display the result.

# The sequence

The screenshot shows a web browser at <https://localhost:34887/>. The interface includes a top navigation bar with a search bar labeled "Rechercher" and a sidebar on the left with a "File" menu and a directory structure: "Home" (selected), "First", and "2". The main content area displays a file named "Sequence.oms" (0,46KB). The file content is a Scala script defining a sequence of tasks:

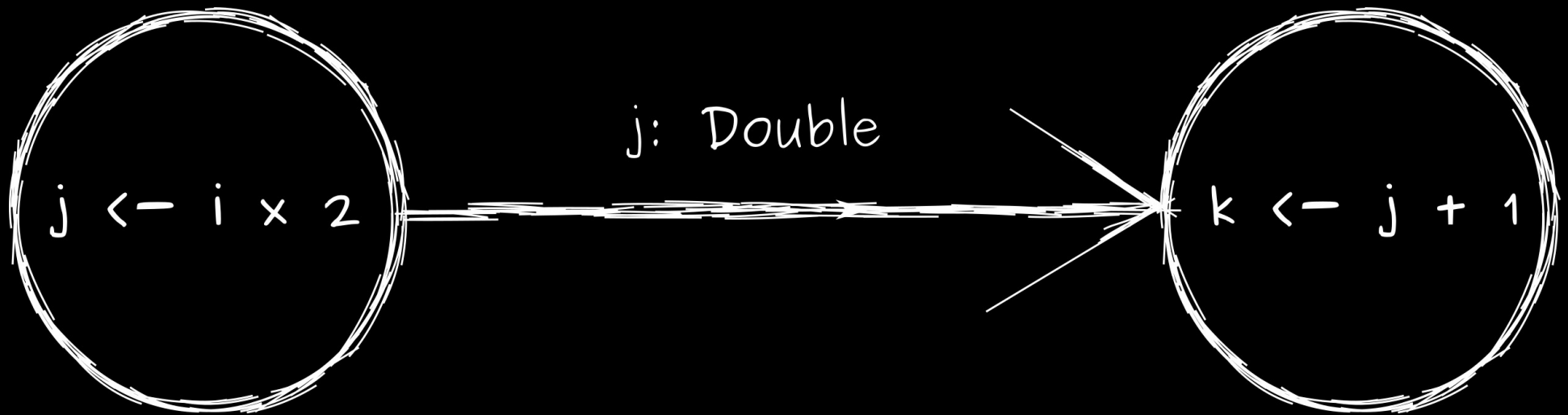
```
1 |
2 | val i = Val[Double]
3 | val j = Val[Double]
4 | val k = Val[Double]
5 |
6 | val by2 =
7 |   ScalaTask("val j = i * 2") set (
8 |     inputs += i,
9 |     outputs += (i, j),
10 |    i := 10.0
11 |  )
12 |
13 | val plus1 =
14 |   ScalaTask("val k = j + 1") set (
15 |     inputs += j,
16 | 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

A "Play" button is visible next to the code. The bottom right corner of the interface features the text "Loving Lobster" 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 transfered from "by2" to "plus1"
21 (by2 hook ToStringHook()) -- (plus1 hook ToStringHook())
22
```

# The sequence

The screenshot displays a web application interface for managing workflow executions. The browser address bar shows `https://localhost:34887/`. The application has a dark sidebar with icons for file management, settings, and other functions. The main content area is titled "Executions" and includes a filter for "Environment error level" set to "ERROR" and an "Output history" dropdown set to "500". A table lists the execution details for "Sequence.oms":

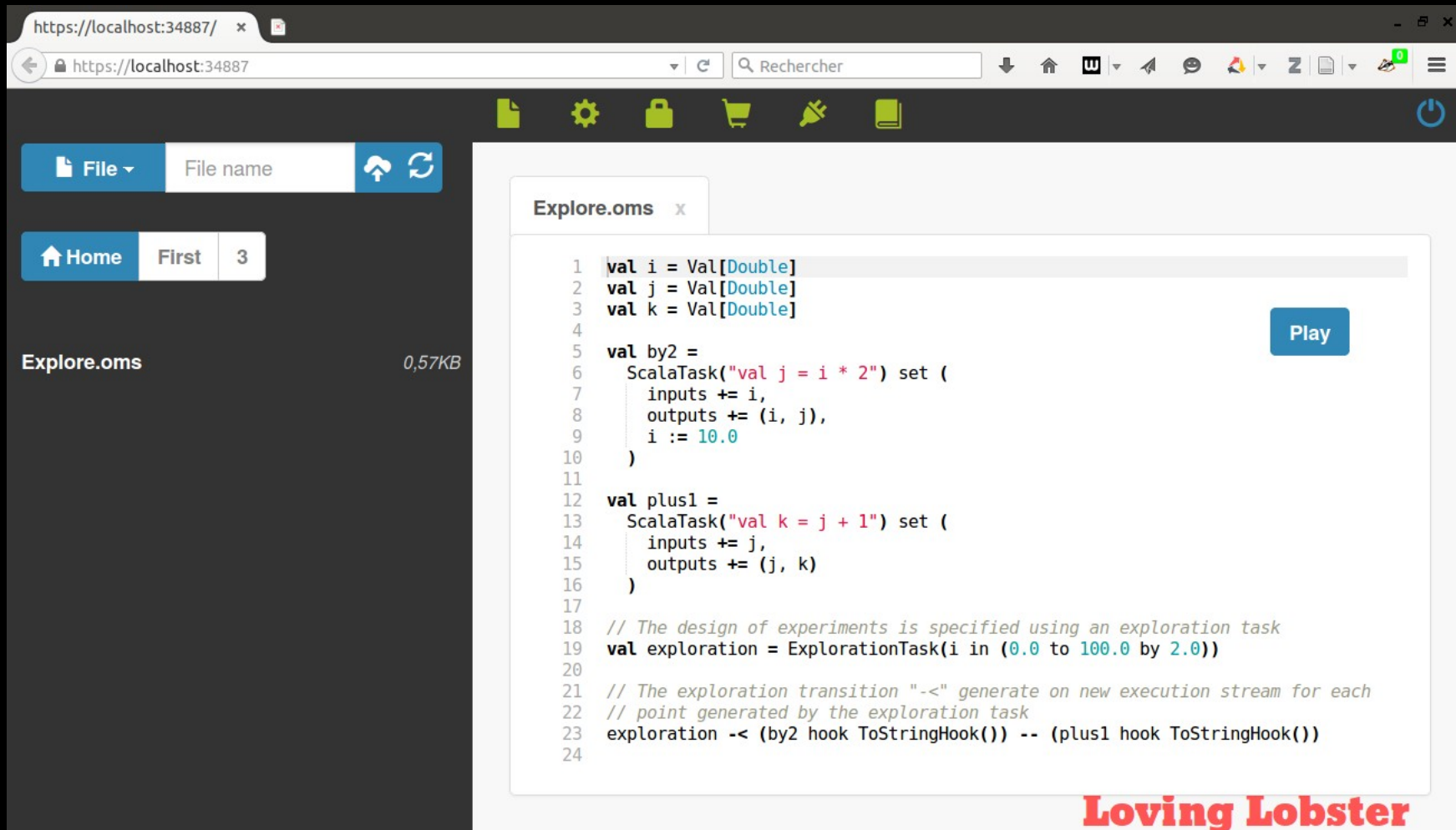
| Execution Name | Timestamp           | Lightning Bolt Icon | Flag Icon | Count | Duration | Status   | Env Icon | More Icon | Delete Icon |
|----------------|---------------------|---------------------|-----------|-------|----------|----------|----------|-----------|-------------|
| Sequence.oms   | 28/9/2015, 13:56:00 | ⚡ 0                 | 🚩         | 2 / 2 | 0:00:03  | finished | Env      | ⋮         | 🗑️          |

Below the table, the workflow definition is shown in a code editor:

```
{i=10.0, j=20.0}  
{j=20.0, k=21.0}
```

Run the workflow.

# The exploration



https://localhost:34887/ x

https://localhost:34887

Rechercher

File File name

Home First 3

Explore.oms 0,57KB

Explore.oms x

```
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())
24
```

Play

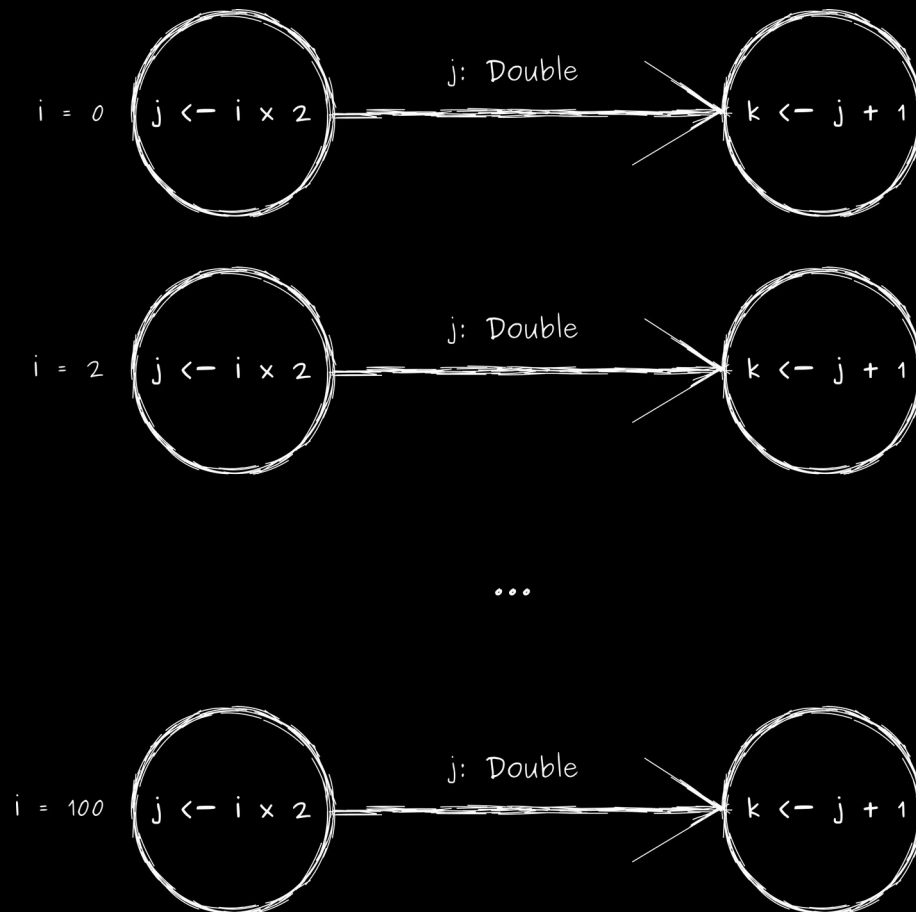
Loving Lobster

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

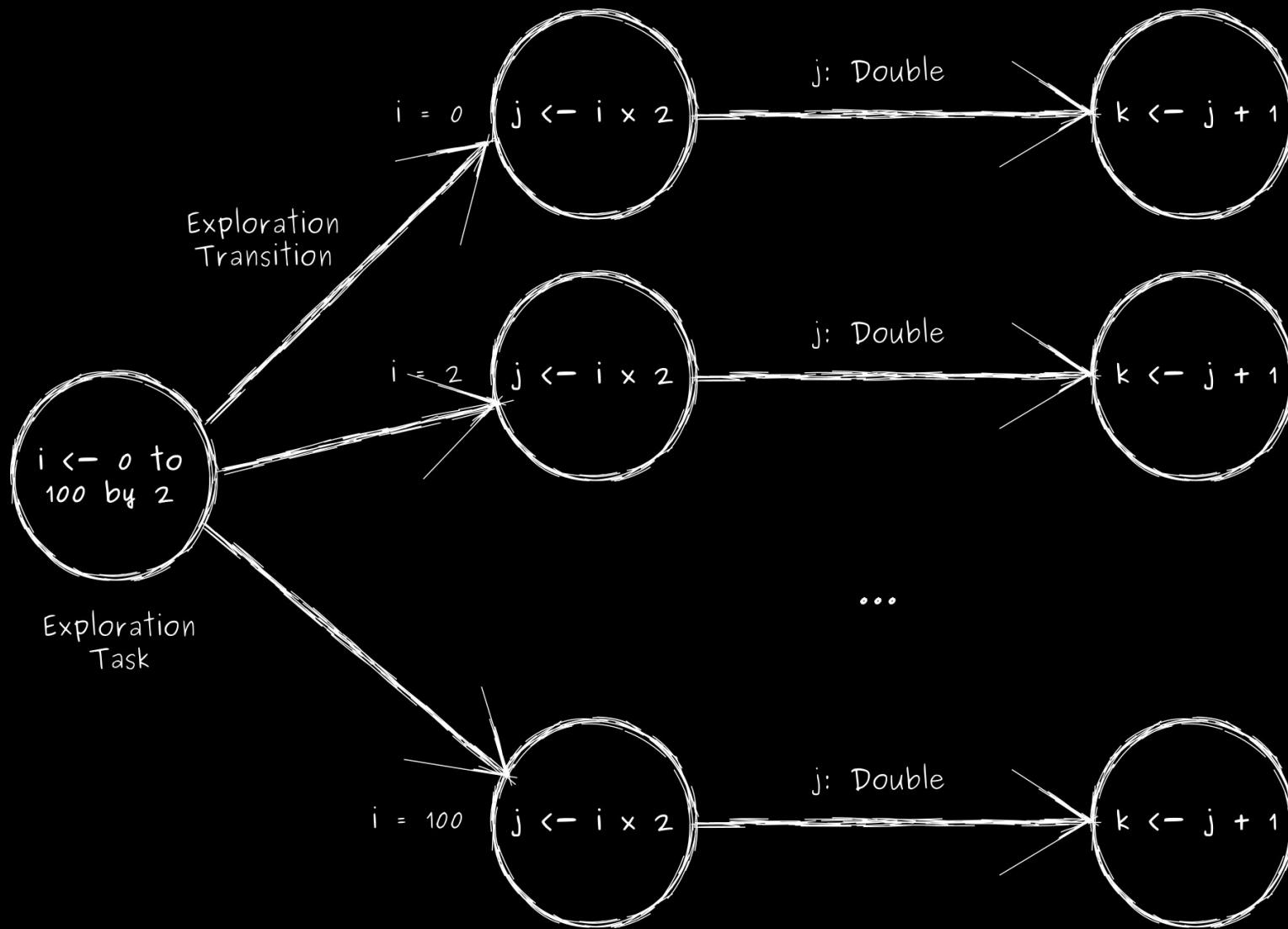


# The exploration

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



# The exploration



Exploration Task + Exploration Transition

# 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

The screenshot shows a web application interface with a dark theme. At the top, there's a browser window with the URL `https://localhost:34887/`. Below the browser, there's a navigation bar with icons for file, settings, folder, shopping cart, and a notification bell. The main content area is titled "Executions" and features a table of execution records. The table has columns for name, date, status, and other metrics. The first row shows "Explore.oms" with a date of "28/9/2015, 14:26:29", a status of "finished", and a value of "103 / 103". Below the table, there's a detailed view of the execution, showing a list of coordinates in the format `{i=x, j=y}`. The coordinates are listed in descending order of `i` and `j` values. The interface also includes a sidebar on the left with a "Home" button and a "Rechercher" search bar.

| Executions  | Environment error level | Output history |
|-------------|-------------------------|----------------|
| Explore.oms | ERROR                   | 500            |

Explore.oms 28/9/2015, 14:26:29 0 103 / 103 0:00:03 finished Env

Explo

- `{i=40.0, j=41.0}`
- `{i=18.0, j=36.0}`
- `{j=36.0, k=37.0}`
- `{i=16.0, j=32.0}`
- `{j=32.0, k=33.0}`
- `{i=14.0, j=28.0}`
- `{j=28.0, k=29.0}`
- `{i=12.0, j=24.0}`
- `{j=24.0, k=25.0}`
- `{i=10.0, j=20.0}`
- `{j=20.0, k=21.0}`
- `{i=8.0, j=16.0}`
- `{j=16.0, k=17.0}`
- `{i=6.0, j=12.0}`
- `{j=12.0, k=13.0}`
- `{i=4.0, j=8.0}`
- `{i=8.0, k=9.0}`

Run it

# store results in files

The screenshot shows a web browser at <https://localhost:34887/> displaying the OpenMOLÉ interface. On the left, a sidebar shows a file explorer with a 'File' button and a 'File name' input field. Below this, there are tabs for 'Home', 'First', and '4'. The '4' tab is selected, and it shows a file named 'Store.oms' with a size of 0,64KB. The main area displays the content of 'Store.oms', which is a Scala script. The script defines variables `i`, `j`, and `k` as `Val[Double]`. It then defines two tasks: `by2` and `plus1`. `by2` is a `ScalaTask` that sets `inputs += i`, `outputs += (i, j)`, and `i := 10.0`. `plus1` is a `ScalaTask` that sets `inputs += j` and `outputs += (j, k)`. The script also defines an `exploration` task that runs `by2` and `plus1` in parallel. Finally, it defines two hooks: `by2Hook` and `plus1Hook`, both of which are `AppendToCSVFileHook` objects that append the results of the tasks to CSV files in the `results` directory. A 'Play' button is visible next to the script. At the bottom right, the text 'Loving Lobster' is displayed.

```
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 OpenMOLÉ 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
```

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

Loving Lobster

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.

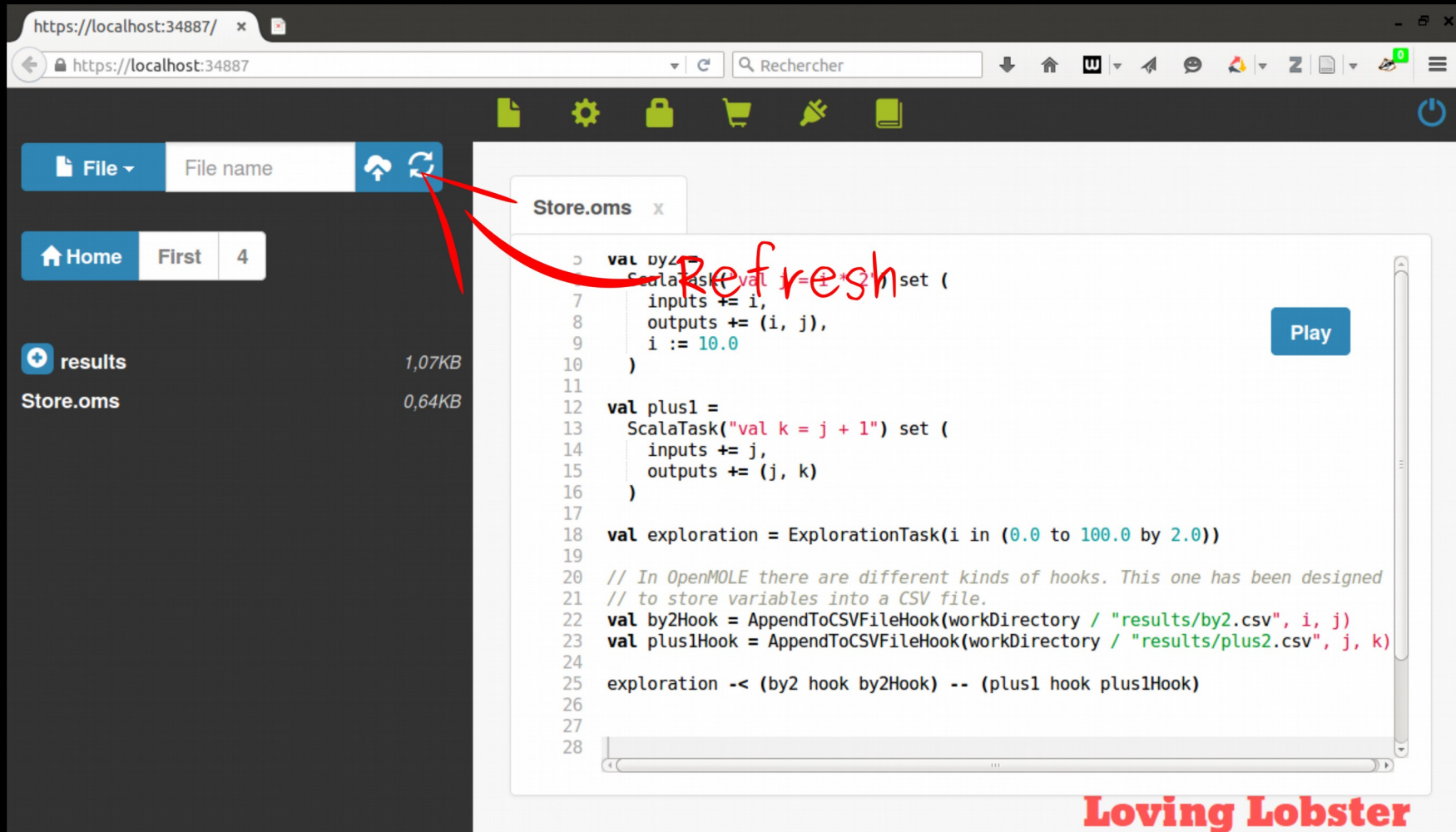
# store results in files

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

The hooks are hooked on the tasks.



# store results in files



The screenshot shows the OpenMole web interface at <https://localhost:34887/>. The interface includes a top navigation bar with icons for file management, settings, and other tools. On the left, there is a sidebar with a 'File' dropdown, a 'File name' input field, and a 'Refresh' button (circular arrow icon). Below this, there are tabs for 'Home', 'First', and '4'. A list of files is shown: 'results' (1,07KB) and 'Store.oms' (0,64KB). The main area displays the content of 'Store.oms', which is a Scala script. The script defines a task 'by2' and a hook 'plus1', and then runs an exploration. A red arrow points from the 'Refresh' button to the 'Store.oms' file name. A red 'Refresh' label is also present over the script content. A 'Play' button is visible on the right side of the script editor.

```
1 val by2 =  
2   ScalaTask("val j = i * 2") set (  
3     inputs += i,  
4     outputs += (i, j),  
5     i := 10.0  
6   )  
7  
8 val plus1 =  
9   ScalaTask("val k = j + 1") set (  
10    inputs += j,  
11    outputs += (j, k)  
12  )  
13  
14 val exploration = ExplorationTask(i in (0.0 to 100.0 by 2.0))  
15  
16 // In OpenMOLE there are different kinds of hooks. This one has been designed  
17 // to store variables into a CSV file.  
18 val by2Hook = AppendToCSVFileHook(workDirectory / "results/by2.csv", i, j)  
19 val plus1Hook = AppendToCSVFileHook(workDirectory / "results/plus2.csv", j, k)  
20  
21 exploration <- (by2 hook by2Hook) -- (plus1 hook plus1Hook)  
22  
23  
24  
25  
26  
27  
28
```

**Loving Lobster**

Run it and refresh.

# store results in files

The screenshot shows a web application running on `https://localhost:34887/`. The interface includes a top navigation bar with a search bar labeled "Rechercher" and a sidebar on the left with a "File" menu and a "Home" button. The main content area displays a file explorer with two files: `by2.csv` (0,52KB) and `plus2.csv` (0,55KB). The `by2.csv` file is selected, and its contents are displayed in a table with 24 rows. The table has two columns, `i` and `j`, and the values are as follows:

| 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 |
| 54.0  | 108.0 |
| 52.0  | 104.0 |

The text "Loving Lobster" is visible in the bottom right corner of the interface.

Results have been stored in the "results" folder.

# store results in files

The screenshot shows a web application interface. At the top, the browser address bar displays `https://localhost:34887/`. Below the browser, there is a navigation bar with a 'File' dropdown, a 'File name' input field, and icons for upload and refresh. A sidebar on the left contains a 'Home' button, a breadcrumb trail '... 4 results', and a list of files: 'by2.csv' (0,52KB) and 'plus2.csv' (0,55KB). A red arrow points from the 'Download' icon next to 'by2.csv' to the 'Download' text. The main content area shows a table with 24 rows of data. The first row is highlighted. The table has two columns: the first column contains integers from 1 to 24, and the second column contains pairs of floating-point numbers separated by a comma. The data in the second column starts with '100.0,200.0' and decreases by 2.0 in the first number and 2.0 in the second number for each subsequent row. A red arrow points from the 'Download' text to the 'Download' icon next to 'by2.csv'. The footer of the page shows the URL `https://localhost:34887/downloadFile?path=projects/First/4/results/by2.csv` and the text 'Loving Lobster'.

Store.oms x by2.csv x

|    |             |
|----|-------------|
| 1  | 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  |

Download

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

Loving Lobster

You can download it.

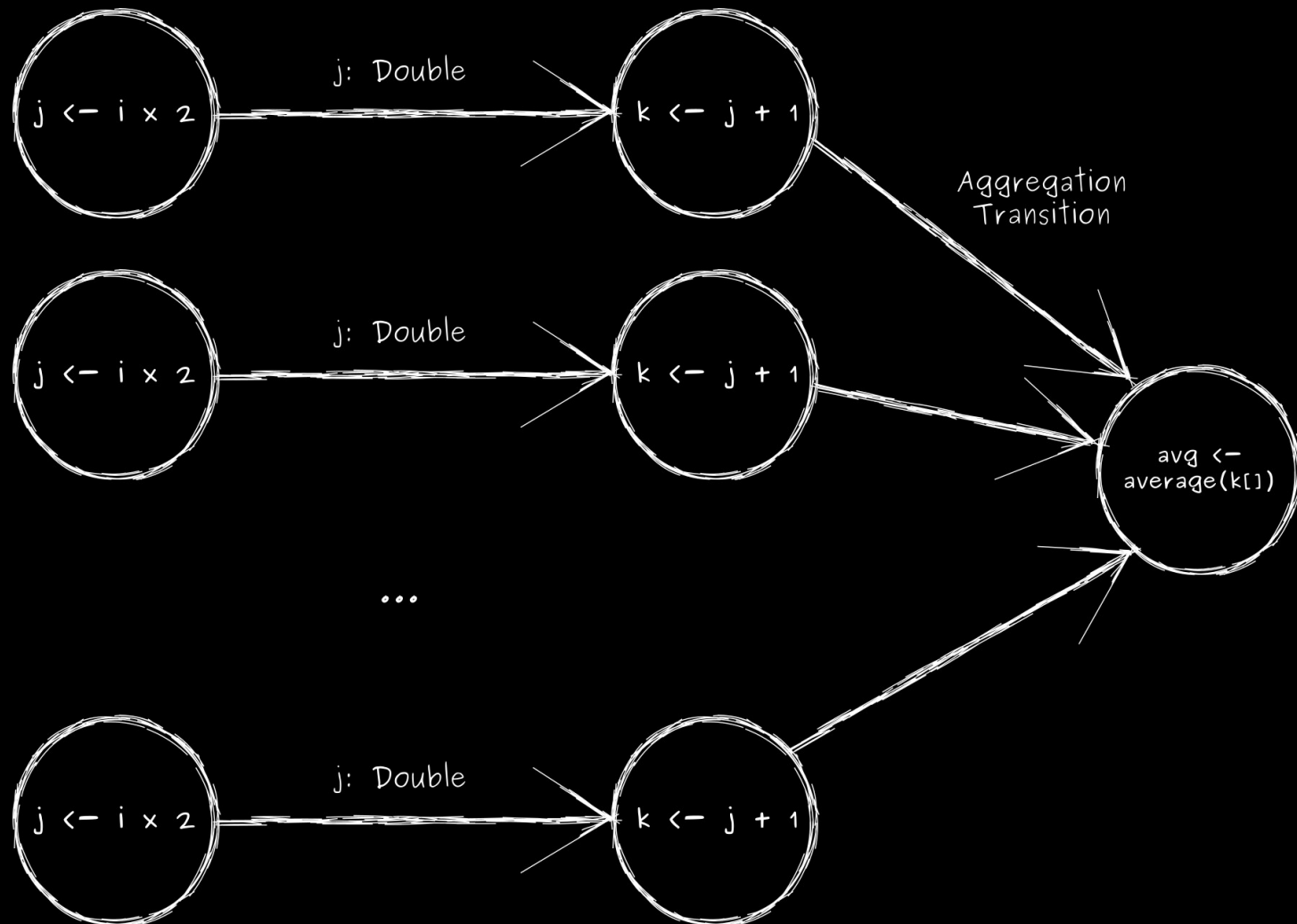
# Aggregate

The screenshot shows a web browser at <https://localhost:34887/>. The interface includes a top navigation bar with icons for file management, settings, and other tools. Below this is a sidebar with a 'File' dropdown, a 'File name' input field, and a directory structure with 'Home', 'First', and '5'. The main content area displays a file named 'Aggregate.oms' (0.77KB) with a code editor. The code is in Scala and defines variables `i`, `j`, `k`, and `avg` as `Val[Double]`. It then defines tasks `by2` and `plus1` using `ScalaTask` to update inputs and outputs. Finally, it defines an `exploration` task and an `average` task that computes the average value from the exploration results. A 'Play' button is visible next to the code editor. The bottom of the browser shows the URL <https://localhost:34887/#d0769e8d-259e-49e7-9fdf-b360de877905> and a 'Loving Lobster' logo.

```
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,
```

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

The screenshot shows a web browser window with the URL `https://localhost:34887/`. The browser's address bar and search bar are visible. Below the browser window, the Aggregate application interface is displayed. The interface has a dark sidebar on the left with icons for file management, settings, and other functions. The main content area is titled "Executions" and features a table of execution records. The first record is for "Aggregate.oms", dated "28/9/2015, 15:56:38", with a status of "finished" and a value of "{avg=101.0}". The interface also includes a "Environment error level" dropdown menu set to "ERROR" and an "Output history" button.

Executions

Environment error level ERROR Output history 500

|               |                     |     |             |         |          |     |   |    |    |
|---------------|---------------------|-----|-------------|---------|----------|-----|---|----|----|
| Aggregate.oms | 28/9/2015, 15:56:38 | ⚡ 0 | 🚩 104 / 104 | 0:00:04 | finished | Env | ⋮ | ⚙️ | 🗑️ |
|---------------|---------------------|-----|-------------|---------|----------|-----|---|----|----|

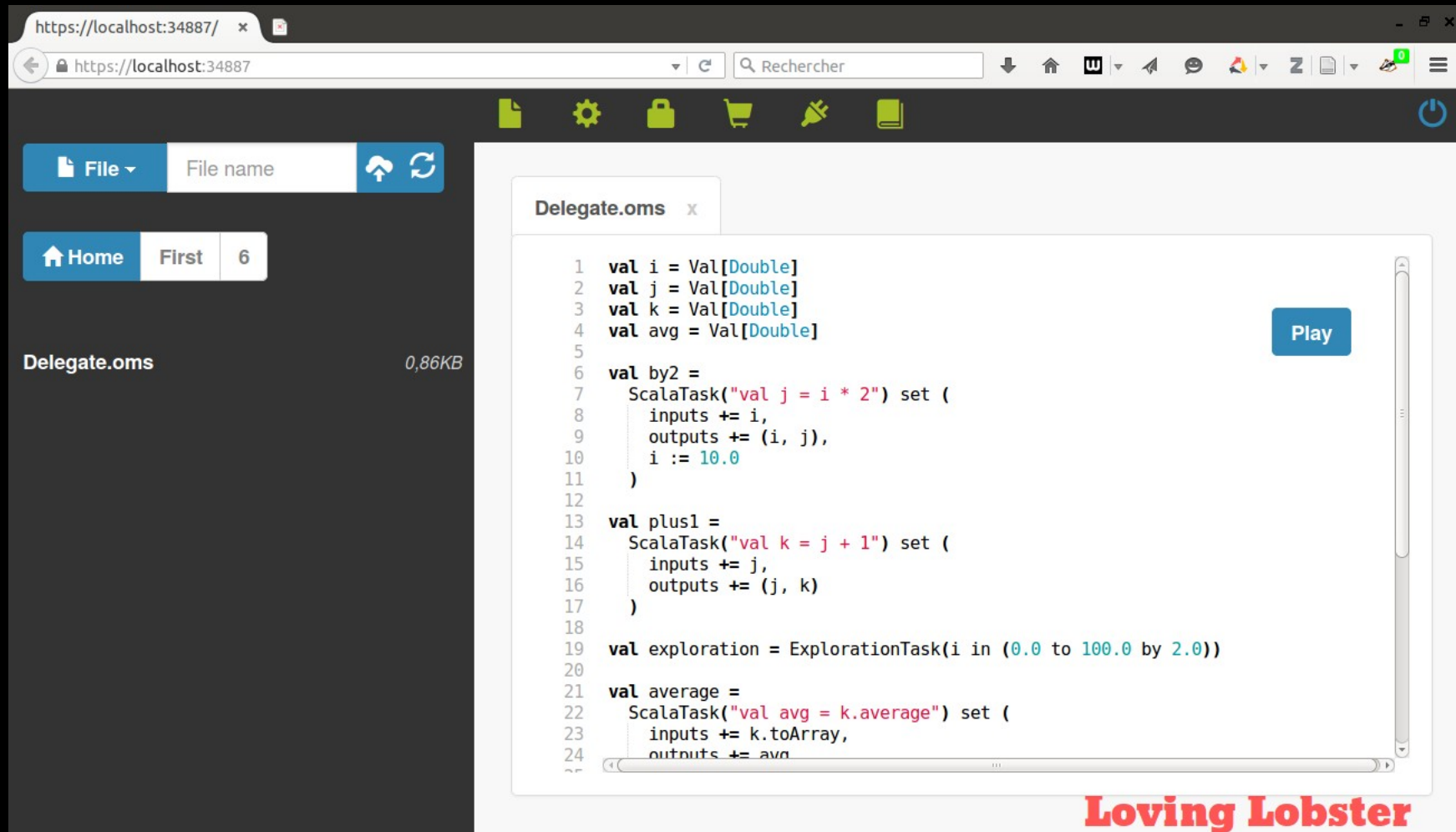
Aggregate

{avg=101.0}

Run it.



# Delegate the execution



The screenshot shows a web browser at <https://localhost:34887/>. The interface includes a file explorer on the left and a code editor on the right.

**File Explorer:**

- Buttons: File, Home, First, 6
- File name: Delegate.oms
- Size: 0,86KB

**Code Editor (Delegate.oms):**

```
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 val average =
22   ScalaTask("val avg = k.average") set (
23     inputs += k.toArray,
24     outputs += avg
```

A **Play** button is located to the right 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)
```

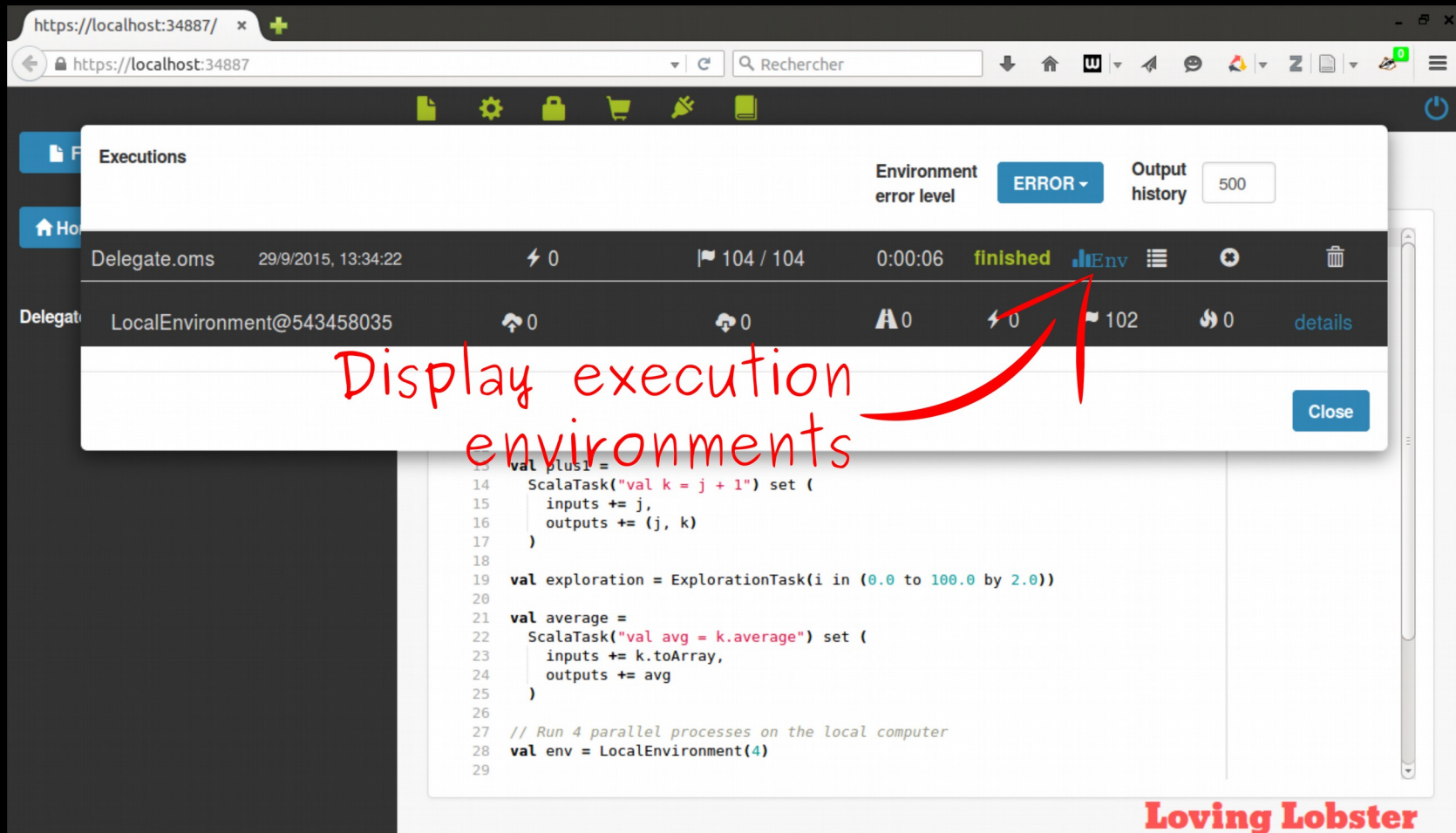
Declare an execution environment:  
4 concurrent local executions

# 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



Environment error level **ERROR** Output history 500

| Job Name                   | Created At          | Duration | Status   | Env | Details |
|----------------------------|---------------------|----------|----------|-----|---------|
| Delegate.oms               | 29/9/2015, 13:34:22 | 0:00:06  | finished | Env |         |
| LocalEnvironment@543458035 |                     |          |          |     | details |

Display execution environments

```
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 planed

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

# Delegate the execution

The screenshot shows a web browser window with the URL `https://zebulon.iscpif.fr:11001`. The interface displays a list of executions. A modal window titled "Executions" is open, showing details for a specific execution. The modal includes a table with the following data:

| Execution Name                 | Date                | Time     | Environment error level | Output history |
|--------------------------------|---------------------|----------|-------------------------|----------------|
| mariusCompletRestreint1959.oms | 26/9/2015, 09:32:58 | 76:11:00 | ERROR                   | 500            |

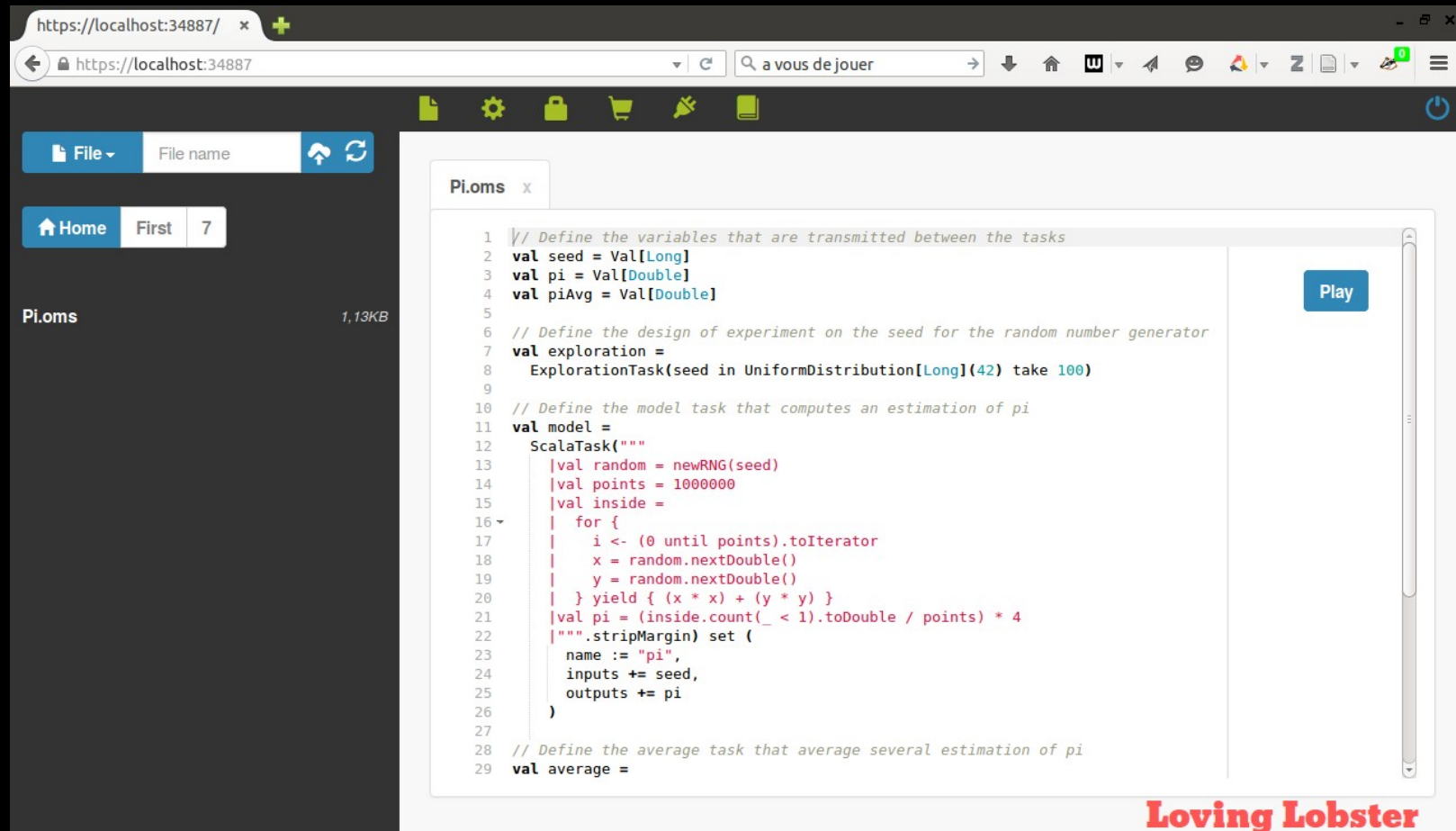
Below the table, there is a section for "EGEnvironment@1746836049" with various statistics: 59 (35.87GB), 13 (1.71GB), 2283, 2597, 66832, and 13123. A "Close" button is visible at the bottom right of the modal.

In the bottom right corner of the interface, there is a logo for "Loving Lobster" featuring a red lobster with the number 5 and the text "Loving Lobster".

OpenMOLE scales!

Millions of jobs, +100GB of data

# And now, play on!



The screenshot shows a web browser interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory named "Pi.oms" with a file named "1,13KB". 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 =
```

A "Play" button is visible in the code editor.

**Loving Lobster**

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.