

Workshop 2 Guide

WORKSHOP SEARCH REASONING

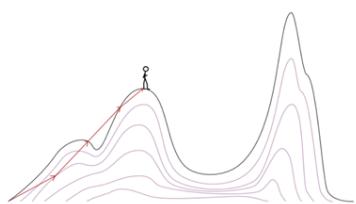
- **KIE OptaPlanner Deep Dive**

- Search Algorithms in Action
- Cloud Computer Balancing

- **KIE OptaPlanner Development – Individual Work**

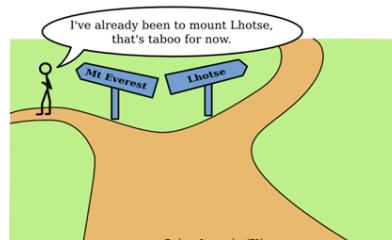
- Choose one OptaPlanner use case, e.g. course curriculum scheduling, airport gate assignment, etc.
- Analyse, adapt, import, and solve using KIE Workbench or programming.

Hill climbing



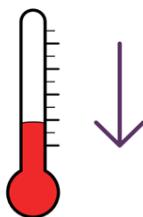
[Hill Climbing \(HC\)](https://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/10/34)

Tabu Search



[Tabu Search \(TS\)](https://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/10/37)

Simulated Annealing



[Simulated Annealing \(SA\)](https://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/10/39)

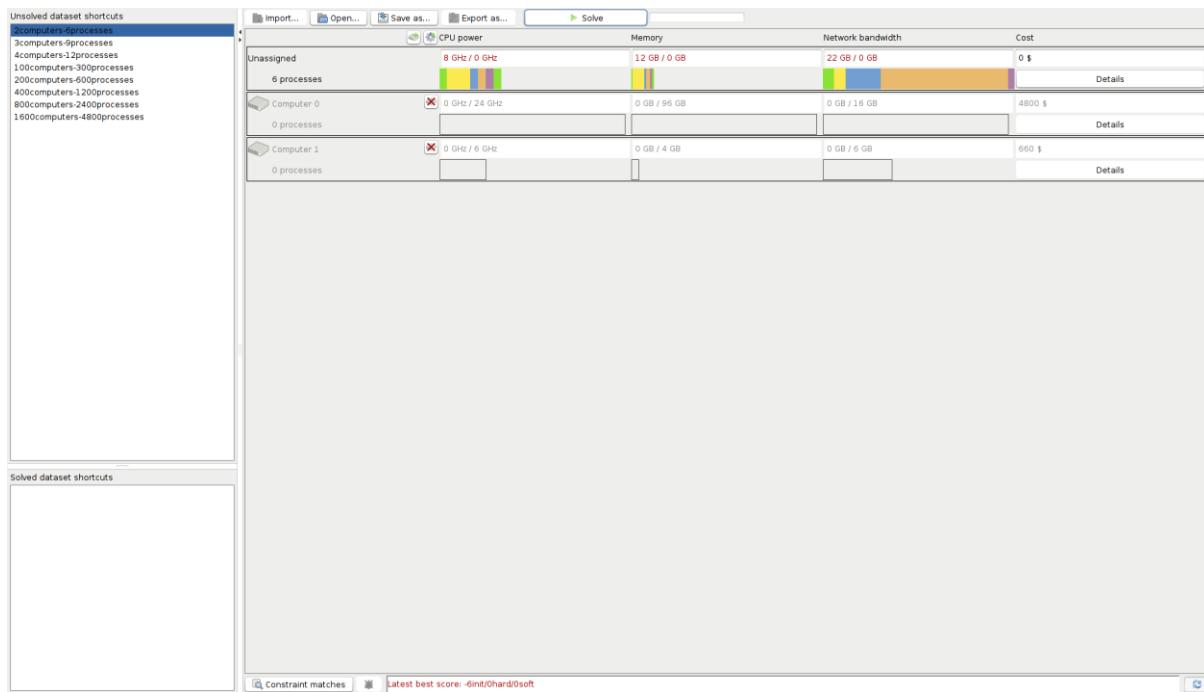
Late acceptance



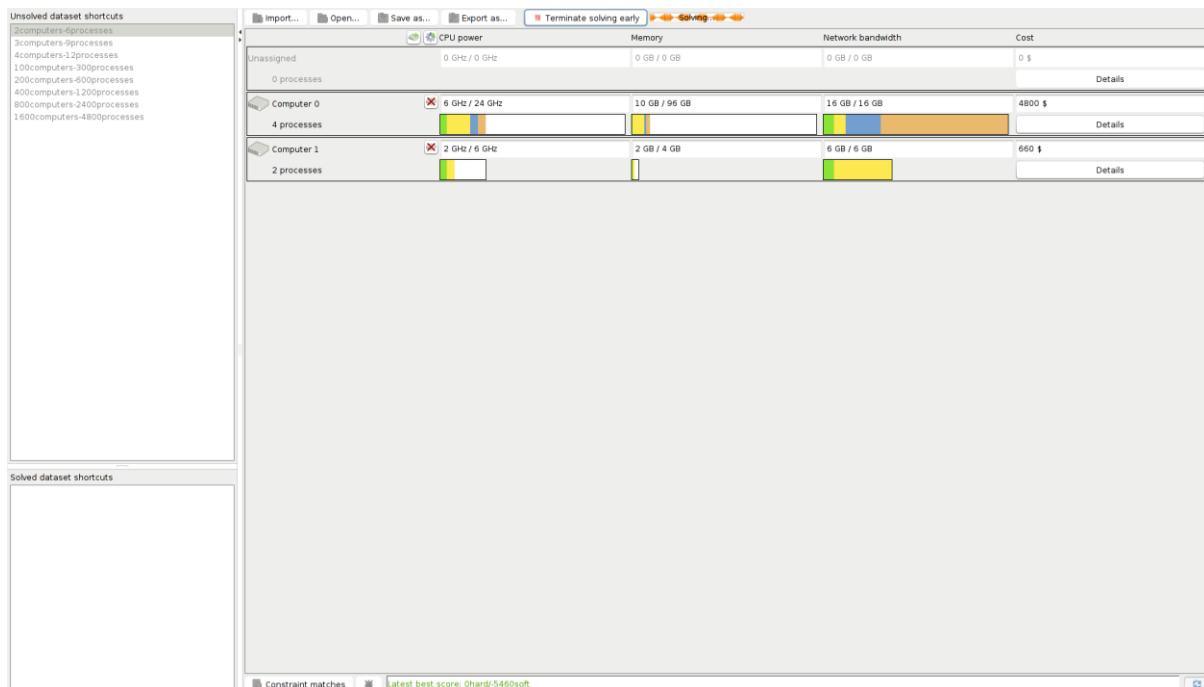
[Late Acceptance Hill Climbing \(LAHC\)](https://www.optaplanner.org/learn/slides/optaplanner-presentation/index.html#/10/41)

Above: Search Algorithms in Action [Demo]

Cloud Computer Balancing [Demo]



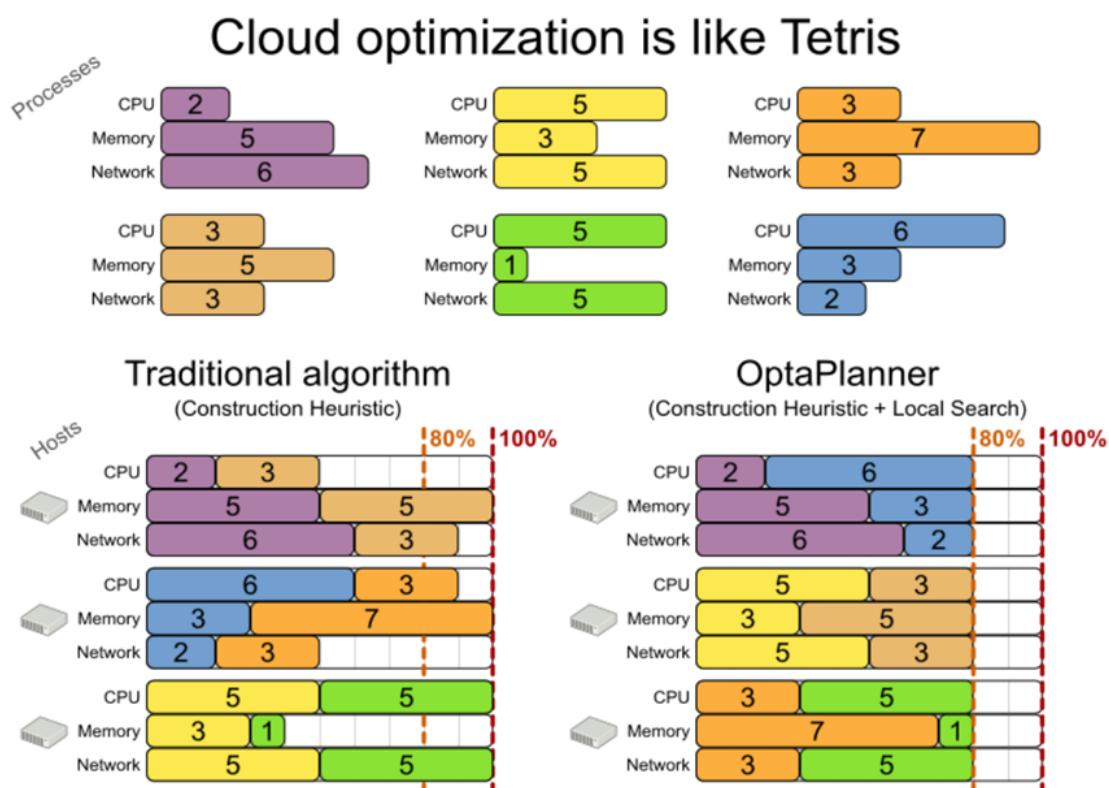
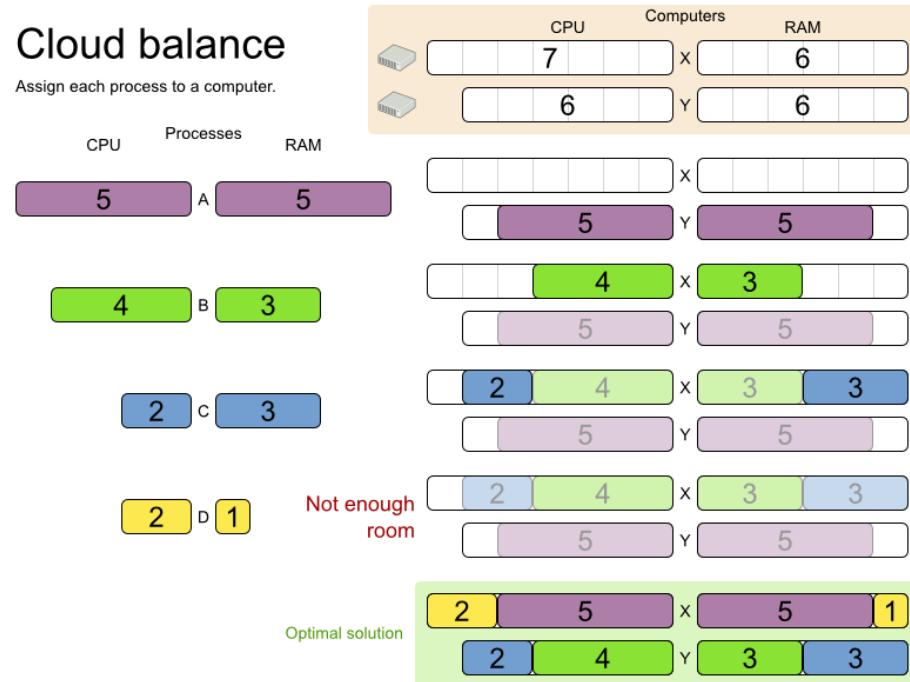
Above: initial solver states



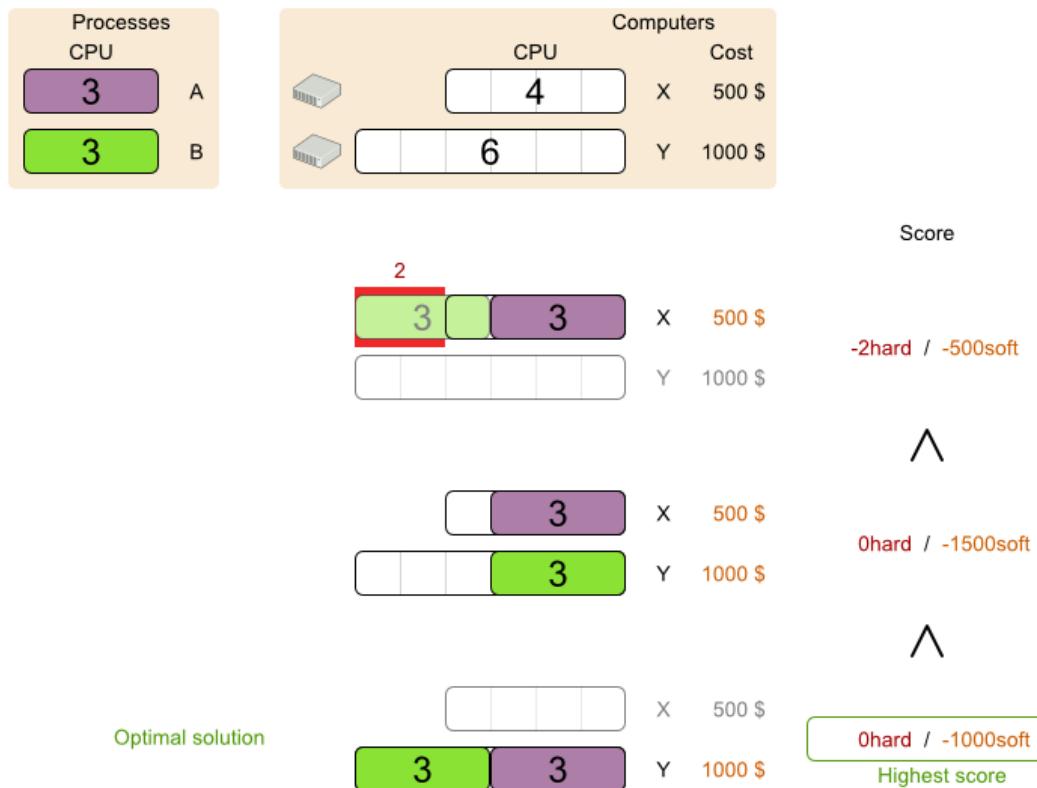
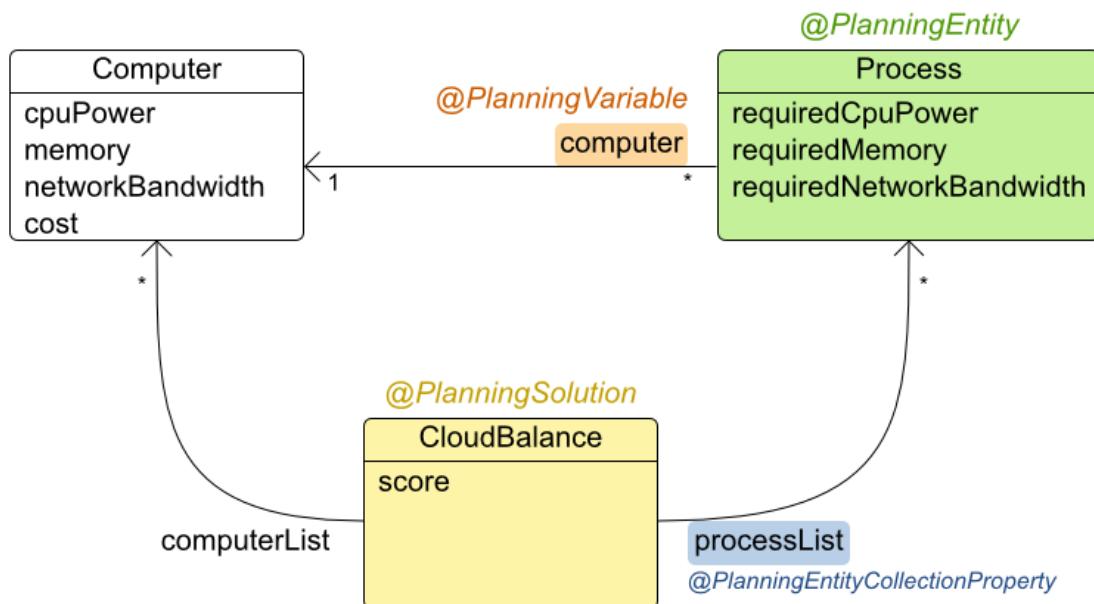
Above: best solution

Workshop 2.1 [Individual]

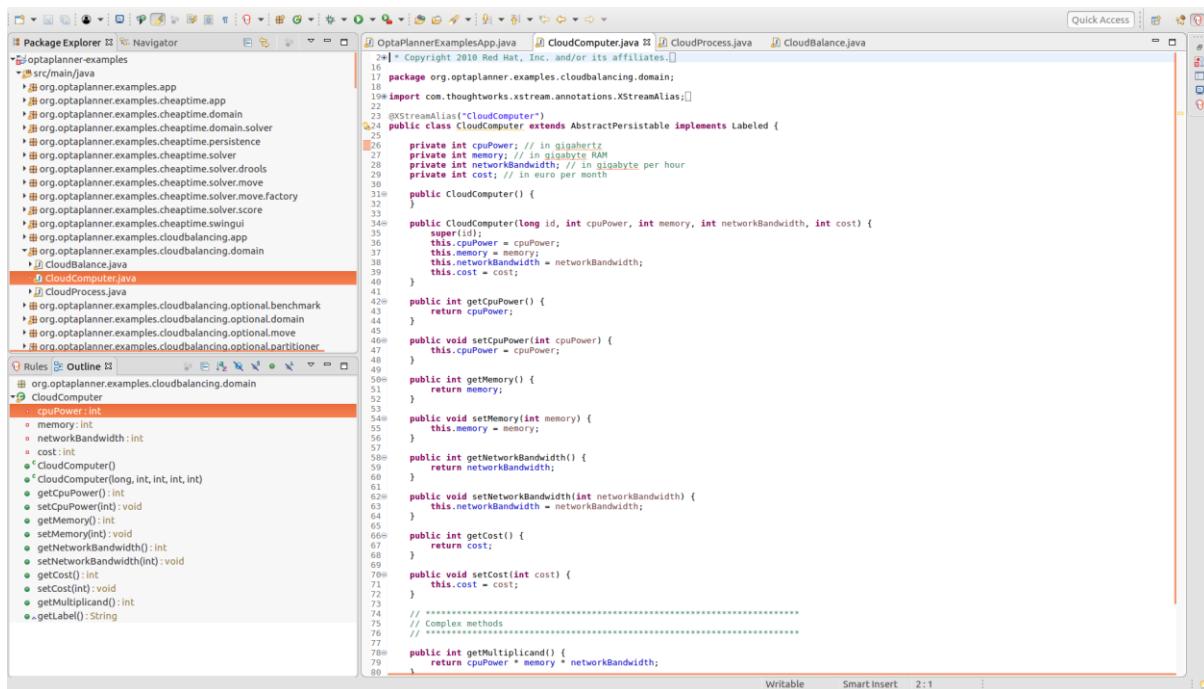
Analyze and execute cloud computer balancing system using both Eclipse and KIE Workbench.



Cloud balance class diagram



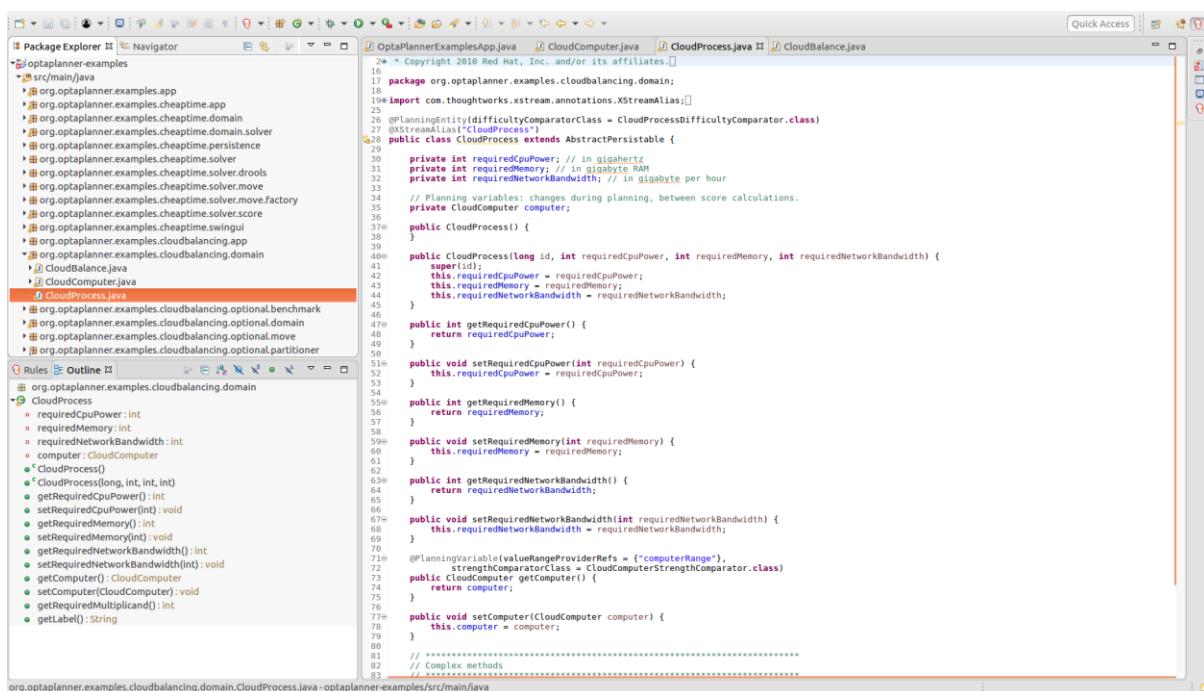
Solver in Java / Eclipse



```


24 * Copyright 2010 Red Hat, Inc. and/or its affiliates.
16
17 package org.optaplanner.examples.cloudbalancing.domain;
18
19 import com.thoughtworks.xstream.annotations.XStreamAlias;
20
21 @XStreamAlias("CloudComputer")
22 public class CloudComputer extends AbstractPersistable implements Labeled {
23
24     private int cpuPower; // in gigahertz
25     private int memory; // in gigabyte RAM
26     private int networkBandwidth; // in gigabyte per hour
27     private int cost; // in euro per month
28
29     public CloudComputer() {
30
31     }
32
33     public CloudComputer(long id, int cpuPower, int memory, int networkBandwidth, int cost) {
34         super(id);
35         this.cpuPower = cpuPower;
36         this.memory = memory;
37         this.networkBandwidth = networkBandwidth;
38         this.cost = cost;
39     }
40
41     public int getCPUPower() {
42         return cpuPower;
43     }
44
45     public void setCPUPower(int cpuPower) {
46         this.cpuPower = cpuPower;
47     }
48
49     public int getMemory() {
50         return memory;
51     }
52
53     public void setMemory(int memory) {
54         this.memory = memory;
55     }
56
57     public int getNetworkBandwidth() {
58         return networkBandwidth;
59     }
60
61     public void setNetworkBandwidth(int networkBandwidth) {
62         this.networkBandwidth = networkBandwidth;
63     }
64
65     public int getCost() {
66         return cost;
67     }
68
69     public void setCost(int cost) {
70         this.cost = cost;
71     }
72
73     // **** Complex methods ****
74     // **** Complex methods ****
75     public int getMultiplicand() {
76         return cpuPower * memory * networkBandwidth;
77     }
78
79 }
80 

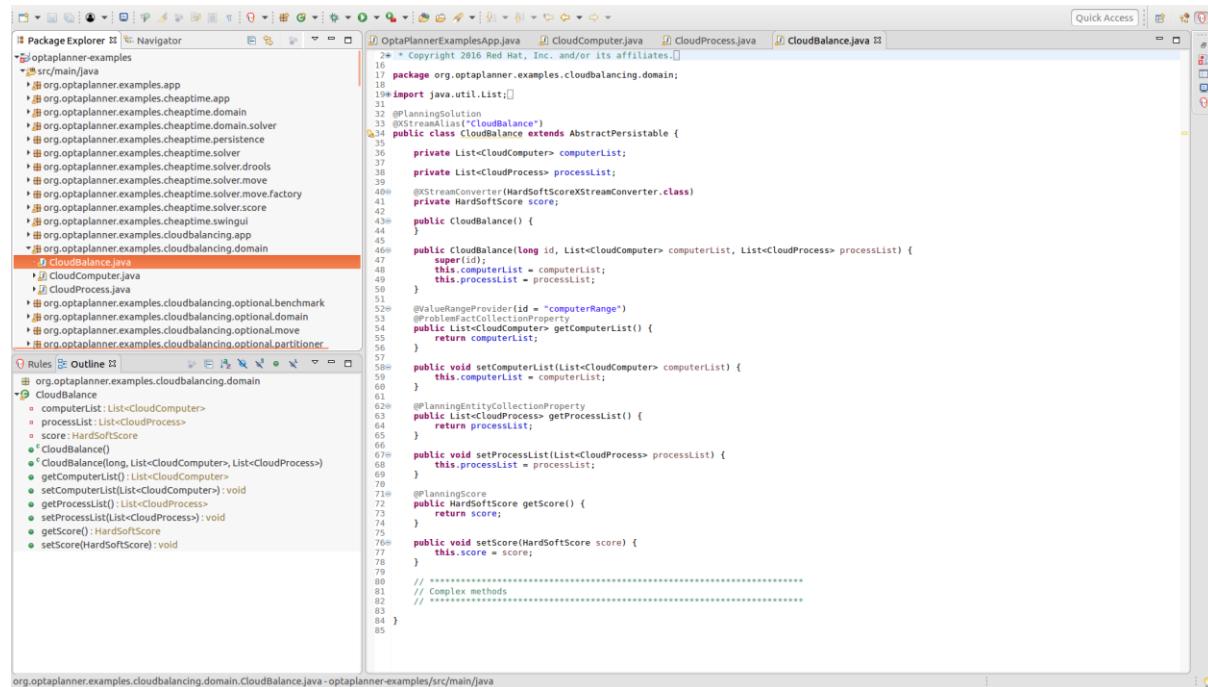
```



```


24 * Copyright 2010 Red Hat, Inc. and/or its affiliates.
17
18 package org.optaplanner.examples.cloudbalancing.domain;
19
20 import com.thoughtworks.xstream.annotations.XStreamAlias;
21
22 @PlanningEntity(difficultyComparatorClass = CloudProcessDifficultyComparator.class)
23 @XStreamAlias("CloudProcess")
24 public class CloudProcess extends AbstractPersistable {
25
26     private int requiredCPUPower; // in gigahertz
27     private int requiredMemory; // in gigabyte RAM
28     private int requiredNetworkBandwidth; // in gigabyte per hour
29
30     // Planning variables: changes during planning, between score calculations.
31     private CloudComputer computer;
32
33     public CloudProcess() {
34
35     }
36
37     public CloudProcess(long id, int requiredCPUPower, int requiredMemory, int requiredNetworkBandwidth) {
38         super(id);
39         this.requiredCPUPower = requiredCPUPower;
40         this.requiredMemory = requiredMemory;
41         this.requiredNetworkBandwidth = requiredNetworkBandwidth;
42     }
43
44     public int getRequiredCPUPower() {
45         return requiredCPUPower;
46     }
47
48     public void setRequiredCPUPower(int requiredCPUPower) {
49         this.requiredCPUPower = requiredCPUPower;
50     }
51
52     public int getRequiredMemory() {
53         return requiredMemory;
54     }
55
56     public void setRequiredMemory(int requiredMemory) {
57         this.requiredMemory = requiredMemory;
58     }
59
60     public int getRequiredNetworkBandwidth() {
61         return requiredNetworkBandwidth;
62     }
63
64     public void setRequiredNetworkBandwidth(int requiredNetworkBandwidth) {
65         this.requiredNetworkBandwidth = requiredNetworkBandwidth;
66     }
67
68     @PlanningVariable(valueRangeProviderRefs = {"computerRange"}, strengthComparatorClass = CloudComputerStrengthComparator.class)
69     public CloudComputer getComputer() {
70         return computer;
71     }
72
73     public void setComputer(CloudComputer computer) {
74         this.computer = computer;
75     }
76
77     // **** Complex methods ****
78     // **** Complex methods ****
79
80 }

```

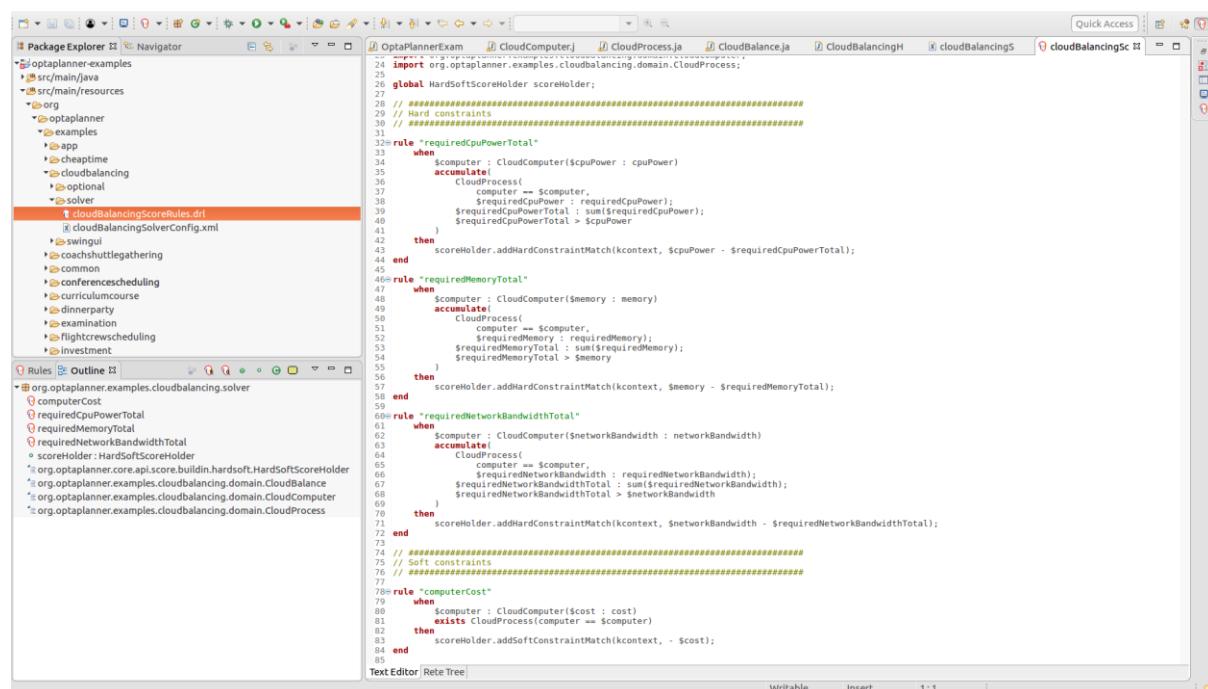


```


1  /*
2  * Copyright 2016 Red Hat, Inc. and/or its affiliates.
3  *
4  * Licensed under the Apache License, Version 2.0 (the "License");
5  * you may not use this file except in compliance with the License.
6  * You may obtain a copy of the License at
7  *
8  *     http://www.apache.org/licenses/LICENSE-2.0
9  *
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16 package org.optaplanner.examples.cloudbalancing.domain;
17
18 import java.util.List;
19
20 @PlanningSolution
21
22 @XStreamAlias("cloudBalance")
23 public class CloudBalance extends AbstractPersistable {
24
25     private List<CloudComputer> computerList;
26
27     private List<CloudProcess> processList;
28
29     private HardSoftScore score;
30
31     public CloudBalance() {
32
33     }
34
35     public CloudBalance(long id, List<CloudComputer> computerList, List<CloudProcess> processList) {
36         super(id);
37         this.computerList = computerList;
38         this.processList = processList;
39     }
40
41     @ValueRangeProviderId("computerRange")
42     @ProblemFactCollectionProperty
43     public List<CloudComputer> getComputerList() {
44         return computerList;
45     }
46
47     public void setComputerList(List<CloudComputer> computerList) {
48         this.computerList = computerList;
49     }
50
51     @PlanningEntityCollectionProperty
52     public List<CloudProcess> getProcessList() {
53         return processList;
54     }
55
56     public void setProcessList(List<CloudProcess> processList) {
57         this.processList = processList;
58     }
59
60     @PlanningScore
61     public HardSoftScore getScore() {
62         return score;
63     }
64
65     public void setScore(HardSoftScore score) {
66         this.score = score;
67     }
68
69     // *****
70     // Complex methods
71     // *****
72 }
73
74
75
76
77
78
79
80
81
82
83
84
85


```

org.optaplanner.examples.cloudbalancing.domain.CloudBalance.java - optaplanner-examples/src/main/java



```


1  package org.optaplanner.examples.cloudbalancing.solver;
2
3  import org.optaplanner.examples.cloudbalancing.domain.CloudProcess;
4
5  global HardSoftScoreHolder scoreHolder;
6
7  // Hard constraints
8  // *****
9
10 rule "requiredCpuPowerTotal"
11 when
12     $computer : CloudComputer($cpuPower : cpuPower)
13     accumulate
14         CloudProcess
15             computer == $computer
16             $cpuPower + requiredCpuPower > requiredCpuPower;
17         $requiredCpuPowerTotal : sum($cpuPower)
18         $RequiredCpuPowerTotal > $cpuPower
19
20 then
21     scoreHolder.addHardConstraintMatch($context, $cpuPower - $requiredCpuPowerTotal);
22 end
23
24 rule "requiredMemoryTotal"
25 when
26     $computer : CloudComputer($memory : memory)
27     accumulate
28         CloudProcess
29             computer == $computer
30             $requiredMemory + requiredMemory > requiredMemory;
31         $requiredMemoryTotal : sum($requiredMemory)
32         $RequiredMemoryTotal > $memory
33
34 then
35     scoreHolder.addHardConstraintMatch($context, $memory - $requiredMemoryTotal);
36 end
37
38 rule "requiredNetworkBandwidthTotal"
39 when
40     $computer : CloudComputer($networkBandwidth : networkBandwidth)
41     accumulate
42         CloudProcess
43             computer == $computer
44             $requiredNetworkBandwidth + requiredNetworkBandwidth > requiredNetworkBandwidth;
45         $requiredNetworkBandwidthTotal : sum($requiredNetworkBandwidth)
46         $RequiredNetworkBandwidthTotal > $networkBandwidth
47
48 then
49     scoreHolder.addHardConstraintMatch($context, $networkBandwidth - $requiredNetworkBandwidthTotal);
50 end
51
52 // Soft constraints
53 // *****
54 rule "computerCost"
55 when
56     $computer : CloudComputer($cost : cost)
57     exists CloudProcess(computer == $computer)
58 then
59     scoreHolder.addSoftConstraintMatch($context, - $cost);
60 end
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85


```

Text Editor | Tree View | Writable | Insert | 1:1

2.5.1. Easy Java Score Configuration

One way to define a score function is to implement the interface `EasyScoreCalculator` in plain Java.

```
<scoreDirectorFactory>
  <easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancing!
</scoreDirectorFactory>
```

Just implement the `calculateScore(Solution)` method to return a `HardSoftScore` instance.

Example 6. CloudBalancingEasyScoreCalculator.java

```
public class CloudBalancingEasyScoreCalculator implements EasyScoreCalculator<CloudBalance> {

    /**
     * A very simple implementation. The double loop can easily be removed by using Maps as shown
     * {@link CloudBalancingMapBasedEasyScoreCalculator#calculateScore(CloudBalance)}.
     */
    public HardSoftScore calculateScore(CloudBalance cloudBalance) {
        int hardScore = 0;
        int softScore = 0;
        for (CloudComputer computer : cloudBalance.getComputerList()) {
            int cpuPowerUsage = 0;
            int memoryUsage = 0;
            int networkBandwidthUsage = 0;
            boolean used = false;

            // Calculate usage
            for (CloudProcess process : cloudBalance.getProcessList()) {
                if (computer.equals(process.getComputer())) {
                    cpuPowerUsage += process.getRequiredCpuPower();
                    memoryUsage += process.getRequiredMemory();
                    networkBandwidthUsage += process.getRequiredNetworkBandwidth();
                    used = true;
                }
            }

            // Hard constraints
            int cpuPowerAvailable = computer.getCpuPower() - cpuPowerUsage;
            if (cpuPowerAvailable < 0) {
                hardScore += cpuPowerAvailable;
            }
            int memoryAvailable = computer.getMemory() - memoryUsage;
            if (memoryAvailable < 0) {
                hardScore += memoryAvailable;
            }
            int networkBandwidthAvailable = computer.getNetworkBandwidth() - networkBandwidthUsage;
            if (networkBandwidthAvailable < 0) {
                hardScore += networkBandwidthAvailable;
            }

            // Soft constraints
            if (used) {
                softScore -= computer.getCost();
            }
        }
        return HardSoftScore.valueOf(hardScore, softScore);
    }
}
```

Even if we optimize the code above to use Maps to iterate through the `processList` only once, it is still slow because it does not do [incremental score calculation](#). To fix that, either use incremental Java score calculation or Drools score calculation.

2.5.2. Drools Score Configuration

Drools score calculation uses incremental calculation, where every score constraint is written as one or more score rules.

Using the Drools rule engine for score calculation, allows you to integrate with other Drools technologies, such as decision tables (XLS or web based), the KIE Workbench, ...

Prerequisite To use the Drools rule engine as a score function, simply add a `scoreDrl` resource in the classpath:

```
<scoreDirectorFactory>
  <scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl</scoreDrl>
</scoreDirectorFactory>
```

1. We want to make sure that all computers have enough CPU, RAM and network bandwidth to support all their processes, so we make these hard constraints:

Example 7. cloudBalancingScoreRules.drl - Hard Constraints

```
...
import org.optaplanner.examples.cloudbalancing.domain.CloudBalance;
import org.optaplanner.examples.cloudbalancing.domain.CloudComputer;
import org.optaplanner.examples.cloudbalancing.domain.CloudProcess;

global HardSoftScoreHolder scoreHolder;

// #####
// Hard constraints
// #####
rule "requiredCpuPowerTotal"
when
  $computer : CloudComputer($cpuPower : cpuPower)
  accumulate(
    CloudProcess(
      computer == $computer,
      $requiredCpuPower : requiredCpuPower);
    $requiredCpuPowerTotal : sum($requiredCpuPower);
    $requiredCpuPowerTotal > $cpuPower
  )
then
  scoreHolder.addHardConstraintMatch(kcontext, $cpuPower - $requiredCpuPowerTotal);
end

rule "requiredMemoryTotal"
...
end

rule "requiredNetworkBandwidthTotal"
...
end
```

2. If those constraints are met, we want to minimize the maintenance cost, so we add that as a soft constraint:

Example 8. cloudBalancingScoreRules.drl - Soft Constraints

```
// #####
// Soft constraints
// #####
rule "computerCost"
when
  $computer : CloudComputer($cost : cost)
  exists CloudProcess(computer == $computer)
then
  scoreHolder.addSoftConstraintMatch(kcontext, - $cost);
```

WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

Example 4. CloudBalancingHelloWorld.java

```

public class CloudBalancingHelloWorld {

    public static void main(String[] args) {
        // Build the Solver
        SolverFactory<CloudBalance> solverFactory = SolverFactory.createFromXmlResource(
            "org/optaplanner/examples/cloudbalancing/solver/cloudBalancingSolverConfig.xml"
        Solver<CloudBalance> solver = solverFactory.buildSolver();

        // Load a problem with 400 computers and 1200 processes
        CloudBalance unsolvedCloudBalance = new CloudBalancingGenerator().createCloudBalance(400);

        // Solve the problem
        CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);

        // Display the result
        System.out.println("\nSolved cloudBalance with 400 computers and 1200 processes:\n" +
            + toDisplayString(solvedCloudBalance));
    }

    ...
}

```

WORKSHOP SEARCH REASONING

KIE OptaPlanner Deep Dive – Cloud Computer Balancing

The solver configuration file determines how the solving process works; it is considered a part of the code. The file is named `cloudBalancingSolverConfig.xml`.

Example 5. cloudBalancingSolverConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<solver>
    <!-- Domain model configuration -->
    <scanAnnotatedClasses/>

    <!-- Score configuration -->
    <scoreDirectorFactory>
        <easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.score.CloudBalanceScoreCalculator</easyScoreCalculatorClass>
        <!--<scoreDrl>org/optaplanner/examples/cloudbalancing/solver/cloudBalancingScoreRules.drl</scoreDrl>
    </scoreDirectorFactory>

    <!-- Optimization algorithms configuration -->
    <termination>
        <secondsSpentLimit>30</secondsSpentLimit>
    </termination>
</solver>

```

```

package org.optaplanner.examples.cloudbalancing.app;
import org.optaplanner.core.api.solver.Solver;
public class CloudBalancingHelloWorld {
    public static void main(String[] args) {
        // Build the solver
        SolverFactory<CloudBalance> solverFactory = SolverFactory.createFromXmlResource(
                "org/optaplanner/examples/cloudbalancing/solver/cloudBalancingSolverConfig.xml");
        Solver<CloudBalance> solver = solverFactory.buildSolver();
        // Load a problem with 400 computers and 1200 processes
        CloudBalance unsolvedCloudBalance = new CloudBalancingGenerator().createCloudBalance(400, 1200);
        // Solve the problem
        CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);
        // Display the result
        System.out.println("\nSolved cloudBalance with 400 computers and 1200 processes:\n" +
                + toDisplayString(solvedCloudBalance));
    }
    public static String toDisplayString(CloudBalance cloudBalance) {
        StringBuilder displayString = new StringBuilder();
        for (CloudProcess process : cloudBalance.getProcessList()) {
            CloudComputer computer = process.getComputer();
            displayString.append(" -").append(process.getLabel()).append(" - ");
            if (computer == null || null == computer.getLabel()) {
                append("\n");
            }
        }
        return displayString.toString();
    }
}

```

The screenshot shows the Eclipse IDE interface with the Package Explorer and Navigator tabs visible. The file `cloudBalancingSolverConfig.xml` is selected in the Package Explorer. The right-hand pane displays the XML configuration file content.

```

<?xml version="1.0" encoding="UTF-8"?
<environmentMode>FULL_ASSERT</environmentMode>
<moveThreadCount>AUTO</moveThreadCount>
<domainModelConfiguration>
<org.optaplanner.examples.cloudbalancing>
<scoreConfiguration>
<scoreDirectorFactory>
<initializingScoreTrend>
<minutesSpentLimit>2
</minutesSpentLimit>
</initializingScoreTrend>
<termination>
<minutesSpentLimit>2
</minutesSpentLimit>
</termination>
<constructionHeuristicType>FIRST_FIT_DECREASING</constructionHeuristicType>
<constructionHeuristic>
<changeMoveSelector>
<swapMoveSelector/>
<pillarChangeMoveSelector/>
<pillarSwapMoveSelector/>
<unionMoveSelector/>
<acceptor>
<entityTabuSize>7</entityTabuSize>
</acceptor>
<forager>
<acceptedCountLimit>1000</acceptedCountLimit>
</forager>
<localSearch>
<alternativePowerTweakedOptimizationAlgorithmsConfiguration>
<partitionedSearch/>

```

	Name	Size	Type	Modified	Accessed	Owner	Group	Permissions	MIME Type
Home	data	22 items	Folder	Sep 26	10:33	Me	iss-user	drwxrwxr-x	inode/directory
Desktop	src	2 items	Folder	Sep 26	10:33	Me	iss-user	drwxrwxr-x	inode/directory
Documents	main	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Downloads	java	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Music	resources	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Pictures	org	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Videos	optaplanner	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Trash	examples	24 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
st_vn_shared_folder	app	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Network	cheaptim	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Computer	cloudbalancing	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
Connect to Server	optional	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	solver	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	cloudBalancingScoreRules.drl	3.0 kB	Text	Sep 26	10:31	Me	iss-user	-rw-rw-r-	text/x-csrc
	cloudBalancingSolverConfig.xml	2.7 kB	Markup	Sep 26	10:31	Me	iss-user	-rw-rw-r-	application/xml
	swingui	14 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	coachshuttlegathering	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	common	1 item	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	conferencescheduling	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	curriculumcourse	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	dinnerparty	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	examination	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	flightcrewscheduling	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	investment	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	machineresassignment	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	meetingscheduling	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	nqueens	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	nurserostering	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	pas	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	projectjobscheduling	3 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory
	rocktour	2 items	Folder	Sep 26	Nov 26	Me	iss-user	drwxrwxr-x	inode/directory

europedit0.xml < wedding0.xml < cloudBalancingSolverConfig.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- solvers -->
3 <!--environmentMode=FULL_ASSERT</environmentMode><!-- To slowly prove there are no bugs in this code-->
4 <!--moveThreadCount=AUTO</moveThreadCount><!-- To solve faster by saturating multiple CPU cores-->
5 
6 <!--Domain model Configuration -->
7 <!-- scanAnnotatedClasses -->
8 | <!-- package include org.optaplanner.examples.cloudbalancing /packageinclude -->
9 | <!-- scanAnnotatedClass -->
10 |
11 <!-- Score configuration -->
12 <!-- scoreDirectorFactory -->
13 | <!--easyScoreCalculatorClass=org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingEasyScoreCalculator</easyScoreCalculatorClass-->
14 | <!--easyScoreCalculatorOrClass=org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingMapBasedEasyScoreCalculator</easyScoreCalculatorClass-->
15 | <!--incrementalScoreCalculatorClass=org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingIncrementalScoreCalculator</incrementalScoreCalculatorClass-->
16 | <!--scoreDirectorFactory=org.optaplanner.examples.cloudbalancing.solver/cloudBalancingScoreRules.drl</scoreDirectorFactory-->
17 | <!--assertionScoreDirectorFactory=POV</initializingScoreTrend-->
18 | <!--easyScoreDirectorFactory-->
19 | <!--easyScoreCalculatorClass=org.optaplanner.examples.cloudbalancing.optional.score.CloudBalancingMapBasedEasyScoreCalculator</easyScoreCalculatorClass-->
20 | <!--assertionScoreDirectorFactory-->
21 </scoreDirectorFactory>
22 
23 <!-- Optimization algorithms configuration -->
24 <!-- termination -->
25 | <!--minutesSpentLimit>2</minutesSpentLimit-->
26 </termination>
27 
28 <!-- Power tweaked optimization algorithms configuration -->
29 <!--constructionHeuristicType=FIRST_FIT_DECREASING</constructionHeuristicType-->
30 <!--constructionHeuristicType-->
31 <!--localSearch -->
32 | <!--unionMoveSelector-->
33 | <!--changeMoveSelector-->
34 | <!--swapMoveSelector-->
35 | <!--ChangeSwapMoveSelector-->
36 | <!--pillarSwapMoveSelector-->
37 | <!--unionMoveSelector-->
38 <!--acceptor -->
39 | <!--entityTabuSize>7</entityTabuSize-->
40 | <!--acceptor-->
41 | <!--forager-->
42 | <!--forager-->
43 | | <!--acceptedCountLimit>1000</acceptedCountLimit-->
44 | <!--forager-->
45 <!--focalSearch-->
46 
47 <!-- Alternative power tweaked optimization algorithms configuration -->
48 <!--partitionedSearch-->
49 | <!--solutionPartitionerClass=org.optaplanner.examples.cloudbalancing.optional.partitioner.CloudBalancePartitioner</solutionPartitionerClass-->
50 | <!--solutionPartitionerCustomProperties-->
51 | | <!--partCount>4</partCount-->
52 | | <!--minimumProcessListSize>300</minimumProcessListSize-->
53 | <!--solutionPartitionerCustomProperties-->
54 <!--partitionedSearch-->
55 </solver>

```

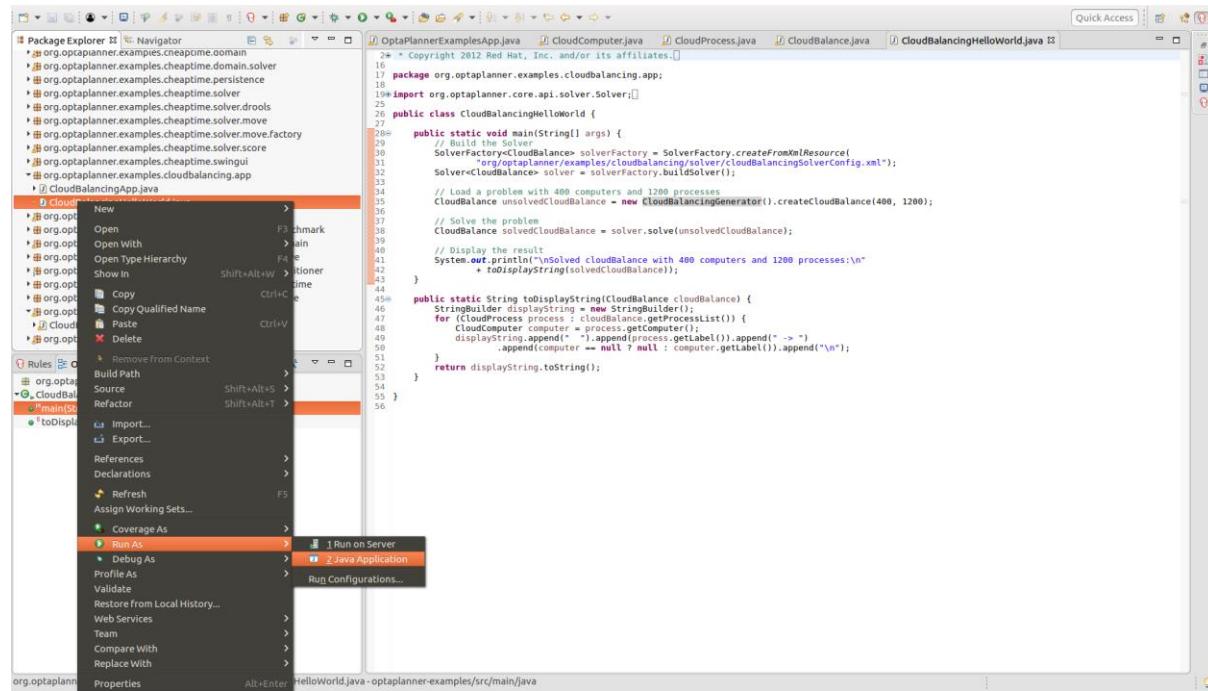
The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure with several Java files under org.optaplanner.examples.cloudbalancing.
- CloudBalancingGenerator.java:** The active code editor window displays the implementation of the CloudBalancingGenerator class. It includes imports for java.io.File, org.optaplanner.examples.cloudbalancing.persistence, and org.optaplanner.examples.common.domain.
- Outline View:** Shows the outline of the current file, including sections like Price, CPU_POWER_PRICES, and MEMORY_PRICES.
- Quick Access:** A toolbar at the top right contains icons for search, refresh, and other common operations.

```
16 package org.optaplanner.examples.cloudbalancing;
17 import java.io.File;
18
19 public class CloudBalancingGenerator extends LoggingMain {
20
21     private static class Price {
22
23         private int hardwareValue;
24         private String description;
25         private int cost;
26
27         private Price(int hardwareValue, String description, int cost) {
28             this.hardwareValue = hardwareValue;
29             this.description = description;
30             this.cost = cost;
31         }
32
33         public int getHardwareValue() {
34             return hardwareValue;
35         }
36
37         public String getDescription() {
38             return description;
39         }
40
41         public int getCost() {
42             return cost;
43         }
44     }
45
46     private static final Price[] CPU_POWER_PRICES = { // in gigahertz
47         new Price(3, "single core 3ghz", 110),
48         new Price(4, "dual core 2ghz", 140),
49         new Price(6, "dual core 3ghz", 180),
50         new Price(8, "quad core 2ghz", 270),
51         new Price(12, "quad core 3ghz", 400),
52         new Price(16, "quad core 4ghz", 1800),
53         new Price(24, "eight core 3ghz", 3000),
54     };
55
56     private static final Price[] MEMORY_PRICES = { // in gigabyte RAM
57         new Price(2, "2 gigabyte", 140),
58         new Price(4, "4 gigabyte", 180),
59         new Price(8, "8 gigabyte", 320),
60         new Price(16, "16 gigabyte", 300),
61         new Price(32, "32 gigabyte", 480),
62         new Price(64, "64 gigabyte", 800),
63         new Price(96, "96 gigabyte", 1000),
64     };
65
66     private static final Price[] NETWORK_BANDWIDTH_PRICES = { // in gigabyte per hour
67         new Price(2, "2 gigabyte", 180),
68         new Price(4, "4 gigabyte", 200),
69         new Price(8, "8 gigabyte", 380),
70         new Price(16, "16 gigabyte", 480),
71         new Price(32, "32 gigabyte", 800),
72         new Price(64, "64 gigabyte", 1800),
73         new Price(96, "96 gigabyte", 1800),
74     };
75
76     private static final int MAXIMUM_REQUIRED_CPU_POWER = 12; // in gigahertz
77     private static final int MAXIMUM_REQUIRED_MEMORY = 32; // in gigabyte RAM
78     private static final int MAXIMUM_REQUIRED_NETWORK_BANDWIDTH = 12; // in gigabyte per hour
79
80     public void checkConfiguration() {
81         writeCloudBalance(1, 1);
82         createCloudBalance(1, 1);
83         determineFileName(1, 1);
84         createCloudBalance(String.valueOf(1), 1);
85         createComputerList(CloudBalance.class);
86         generateComputerWithoutDd();
87     }
88
89     public void writeCloudBalance(int id, int count) {
90         CloudBalance cloudBalance = new CloudBalance(id, count);
91         FileUtil.writeObject(cloudBalance, "src/main/java/org/optaplanner/examples/cloudbalancing/persistence/CloudBalanceGenerator.java");
92     }
93 }
```

The screenshot shows the Eclipse IDE interface with two code editors open. The left editor contains `CloudBalancingGenerator.java` and the right editor contains `CloudComputer.java`. Both files are part of the `OptaPlannerExamplesApp` project. The code in `CloudBalancingGenerator.java` includes imports for various cloud balancing components and logic for generating cloud balances and computer lists. The code in `CloudComputer.java` defines a class with methods for generating cloud computers and determining solution file paths.

```
126 public CloudBalancingGenerator() {
127     solutionFileIO = new XStreamSolutionFileIO<CloudBalance.class>();
128     outputDir = new File(CommonApp.determineDataDir(CloudBalancingApp.DATA_DIR_NAME), "unsolved");
129 }
130 }
131
132 public CloudBalancingGenerator(boolean withoutDao) {
133     if (!withoutDao) {
134         throw new IllegalArgumentException("The parameter withoutDao (" + withoutDao + ") must be true.");
135     }
136     checkConfiguration();
137     solutionFileIO = null;
138     outputDir = null;
139 }
140
141 private void checkConfiguration() {
142     if (CPU_POWER_PRICES.length != MEMORY_PRICES.length || CPU_POWER_PRICES.length != NETWORK_BANDWIDTH_PRICES.length) {
143         throw new IllegalStateException("All price arrays must be equal in length.");
144     }
145 }
146
147 private void writeCloudBalance(int computerListSize, int processListSize) {
148     String fileName = determineFileName(computerListSize, processListSize);
149     File outputFile = new File(outputDir, fileName + ".xml");
150     CloudBalance cloudBalance = createCloudBalance(fileName, computerListSize, processListSize);
151     solutionFileIO.writeCloudBalance(outputFile, cloudBalance);
152     logger.info("Saved: {}", outputFile);
153 }
154
155 public CloudBalance createCloudBalance(int computerListSize, int processListSize) {
156     return createCloudBalance(determineFileName(computerListSize, processListSize),
157                               computerListSize, processListSize);
158 }
159
160 private String determineFileName(int computerListSize, int processListSize) {
161     return computerListSize + "-" + computers + "-" + processes;
162 }
163
164 public CloudBalance createCloudBalance(String inputId, int computerListSize, int processListSize) {
165     random = new Random(47);
166     CloudBalance cloudBalance = new CloudBalance();
167     cloudBalance.setId(inputId);
168     createCloudBalance(cloudBalance, computerListSize);
169     createProcessList(cloudBalance, processListSize);
170     assureComputerCapacityTotalAtLeastProcessesRequiredTotal(cloudBalance);
171     BigInteger possibleSolutionSize = BigInteger.valueOf(cloudBalance.getComputerList().size()).pow(
172         cloudBalance.getProcessList().size());
173     logger.info("CloudBalance {} has {} computers and {} processes with a search space of {}.", inputId,
174               computerListSize, processListSize,
175               possibleSolutionImporter.getFlooredPossibleSolutionSize(possibleSolutionSize));
176     return cloudBalance;
177 }
178
179 private void createComputerList(CloudBalance cloudBalance, int computerListSize) {
180     List<CloudComputer> computerList = new ArrayList<>(computerListSize);
181     for (int i = 0; i < computerListSize; i++) {
182         CloudComputer computerWithoutId = generateComputerWithoutId();
183         computerWithoutId.setId((long) i);
184         computerList.add(computerWithoutId);
185     }
186     cloudBalance.setComputerList(computerList);
187 }
188
189 public CloudComputer generateComputerWithoutId() {
190 }
```



```


1 // * Copyright 2012 Red Hat, Inc. and/or its affiliates.
2
3 package org.optaplanner.examples.cloudbalancing.app;
4
5 import org.optaplanner.core.api.solver.Solver;
6
7 public class CloudBalancingHelloWorld {
8
9     public static void main(String[] args) {
10         // Build the Solver
11         SolverFactory<CloudBalance> solverFactory = SolverFactory.createFromXmlResource(
12             "org/optaplanner/examples/cloudbalancing/server/cloudBalancingSolverConfig.xml");
13         Solver<CloudBalance> solver = solverFactory.buildSolver();
14
15         // Load a problem with 400 computers and 1200 processes
16         CloudBalance unsolvedCloudBalance = new CloudBalancingGenerator().createCloudBalance(400, 1200);
17
18         // Solve the problem
19         CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);
20
21         // Display the result
22         System.out.println("\nSolved cloudBalance with 400 computers and 1200 processes:\n"
23             + toDisplayString(solvedCloudBalance));
24
25     }
26
27     public static String toDisplayString(CloudBalance cloudBalance) {
28         StringBuilder displayString = new StringBuilder();
29         for (CloudProcess process : cloudBalance.getProcessList()) {
30             CloudComputer computer = process.getComputer();
31             displayString.append(" -> ").append(process.getLabel()).append(" -> ");
32             if (computer == null || !computer.getLabel().append("\n"));
33             displayString.append(computer.getLabel());
34         }
35         return displayString.toString();
36     }
37
38     public static void main(String[] args) {
39         // Load a problem with 400 computers and 1200 processes
40         CloudBalance unsolvedCloudBalance = new CloudBalancingGenerator().createCloudBalance(400, 1200);
41
42         // Solve the problem
43         CloudBalance solvedCloudBalance = solver.solve(unsolvedCloudBalance);
44
45         // Display the result
46         System.out.println("\nSolved cloudBalance with 400 computers and 1200 processes:\n"
47             + toDisplayString(solvedCloudBalance));
48
49     }
50
51     public static String toDisplayString(CloudBalance cloudBalance) {
52         StringBuilder displayString = new StringBuilder();
53         for (CloudProcess process : cloudBalance.getProcessList()) {
54             CloudComputer computer = process.getComputer();
55             displayString.append(" -> ").append(process.getLabel()).append(" -> ");
56             if (computer == null || !computer.getLabel().append("\n"));
57             displayString.append(computer.getLabel());
58         }
59         return displayString.toString();
60     }
61
62 }


```

```

terminated>CloudBalancingHelloWorld [Java Application]/usr/lib/jvm/java-8-openjdk-amd64/bin/java [28 Nov 2016, 10:55:44]
12:08:00.848 [main] DEBUG Ls step (216328), time spent (119987), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/2), picked move (CloudProcess-946 (CloudComputer)
12:08:00.849 [main] DEBUG Ls step (216329), time spent (119988), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/5), picked move (CloudProcess-1187 (CloudComputer)
12:08:00.850 [main] DEBUG Ls step (216330), time spent (119988), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/5), picked move (CloudProcess-1188 (CloudComputer)
12:08:00.849 [main] DEBUG Ls step (216332), time spent (119988), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/1), picked move (CloudProcess-689 (CloudComputer)
12:08:00.849 [main] DEBUG Ls step (216333), time spent (119988), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/9), picked move (CloudProcess-219 (CloudComputer)
12:08:00.850 [main] DEBUG Ls step (216334), time spent (119988), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/7), picked move (CloudProcess-172 (CloudComputer)
12:08:00.850 [main] DEBUG Ls step (216335), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/7), picked move (CloudProcess-1068 (CloudComputer)
12:08:00.850 [main] DEBUG Ls step (216336), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/14), picked move (CloudProcess-315 (CloudComputer)
12:08:00.850 [main] DEBUG Ls step (216337), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/14), picked move (CloudProcess-1069 (CloudComputer)
12:08:00.851 [main] DEBUG Ls step (216338), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/4), picked move (CloudProcess-201 (CloudComputer)
12:08:00.851 [main] DEBUG Ls step (216340), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/4), picked move (CloudProcess-1042 (CloudComputer)
12:08:00.851 [main] DEBUG Ls step (216341), time spent (119989), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/11), picked move (CloudProcess-791 (CloudComputer)
12:08:00.852 [main] DEBUG Ls step (216342), time spent (119991), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/11), picked move (CloudProcess-437 (CloudComputer)
12:08:00.852 [main] DEBUG Ls step (216343), time spent (119991), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/11), picked move (CloudProcess-574 (CloudComputer)
12:08:00.853 [main] DEBUG Ls step (216345), time spent (119992), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/28), picked move (CloudProcess-574 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216346), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/17), picked move (CloudProcess-26 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216347), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/22), picked move (CloudProcess-507 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216348), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/22), picked move (CloudProcess-534 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216349), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/11), picked move (CloudProcess-251 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216350), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/22), picked move (CloudProcess-251 (CloudComputer)
12:08:00.854 [main] DEBUG Ls step (216351), time spent (119993), score (0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (1/22), picked move (CloudProcess-251 (CloudComputer)
12:08:00.861 [main] DEBUG Ls step (216352), time spent (120000), score (-0hard/-48126soft), best score (0hard/-48880soft), accepted/selected move count (0/9), picked move (CloudProcess-779 (CloudComputer)
12:08:00.861 [main] INFO Local Search phase (1) ended: time spent (120001), best score (0hard/-48880soft), score calculation speed (10470/sec), step total (216353).
12:08:00.861 [main] INFO Solving ended: time spent (120001), best score (0hard/-48880soft), score calculation speed (12368/sec), phase total (2), environment mode (REPRODUCIBLE).

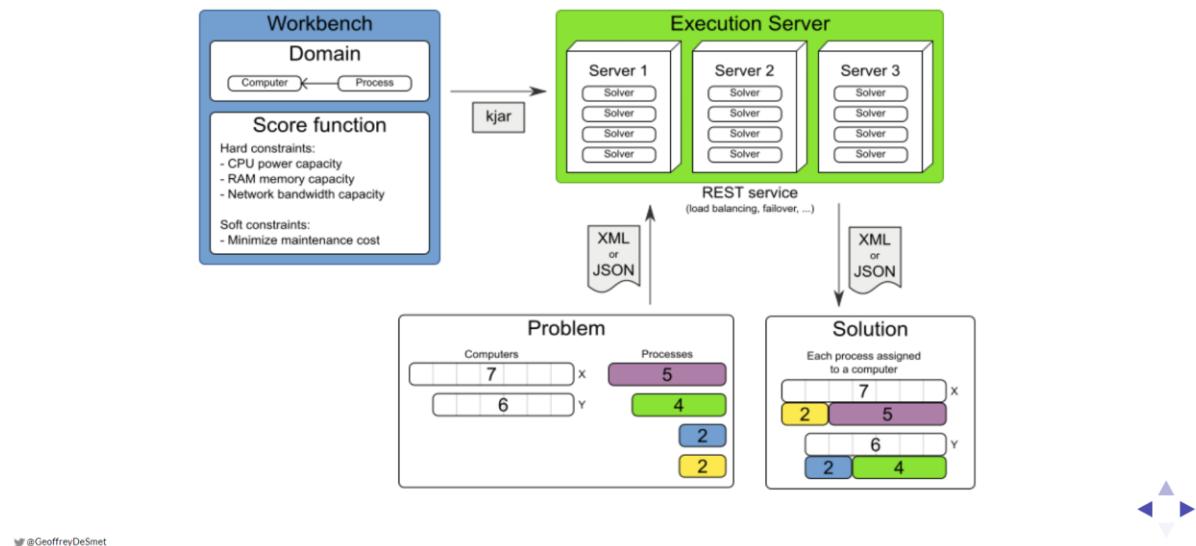
Solved cloudBalance with 400 computers and 1200 processes:
  Process 0 -> Computer 195
  Process 1 -> Computer 137
  Process 2 -> Computer 207
  Process 3 -> Computer 19
  Process 4 -> Computer 16
  Process 5 -> Computer 351
  Process 6 -> Computer 388
  Process 7 -> Computer 68
  Process 8 -> Computer 0
  Process 9 -> Computer 195
  Process 10 -> Computer 295
  Process 11 -> Computer 247
  Process 12 -> Computer 260
  Process 13 -> Computer 99
  Process 14 -> Computer 169
  Process 15 -> Computer 240
  Process 16 -> Computer 125
  Process 17 -> Computer 375
  Process 18 -> Computer 370
  Process 19 -> Computer 119
  Process 20 -> Computer 99
  Process 21 -> Computer 168
  Process 22 -> Computer 91
  Process 23 -> Computer 186
  Process 24 -> Computer 106
  Process 25 -> Computer 191
  Process 26 -> Computer 230
  Process 27 -> Computer 58
  Process 28 -> Computer 293
  Process 29 -> Computer 179
  Process 30 -> Computer 397
  Process 31 -> Computer 249
  Process 32 -> Computer 192

```

Solver in KIE Workbench & Server

OptaPlanner Workbench and Execution Server

Define a use case in the Workbench and then deploy it to the Execution Server to solve it in the cloud.



© Geoffrey De Smet

The screenshot shows the KIE Workbench interface with the title bar "KIE Workbench". The address bar indicates the URL is `localhost:8080/bpm-console/kie-wb.jsp#LibraryPerspective!`. The main navigation bar includes "Assets", "Contributors", "Metrics", and "Settings". Below this is a search bar and a pagination area showing "1-8 of 8" with "Import Asset" and "Add Asset" buttons.

Asset Type	Description	Last modified	Created
cloudScoreRules	DRL	Last modified today	Created today
CloudSolution	Data Objects	Last modified today	Created today
CloudSolutionScoreHolderGlobal	Globals Definitions	Last modified today	Created today
cloudServerConfig	Solver configuration	Last modified today	Created today
Computer	Data Objects	Last modified today	Created today
Process	Data Objects	Last modified today	Created today
SolverTest	Data Objects	Last modified today	Created today
test	Globals Definitions	Last modified today	Created today

Above: try-sample project *OptaCloud_ISS_RS* in KIE Workbench

Above: domain objects based on class diagram

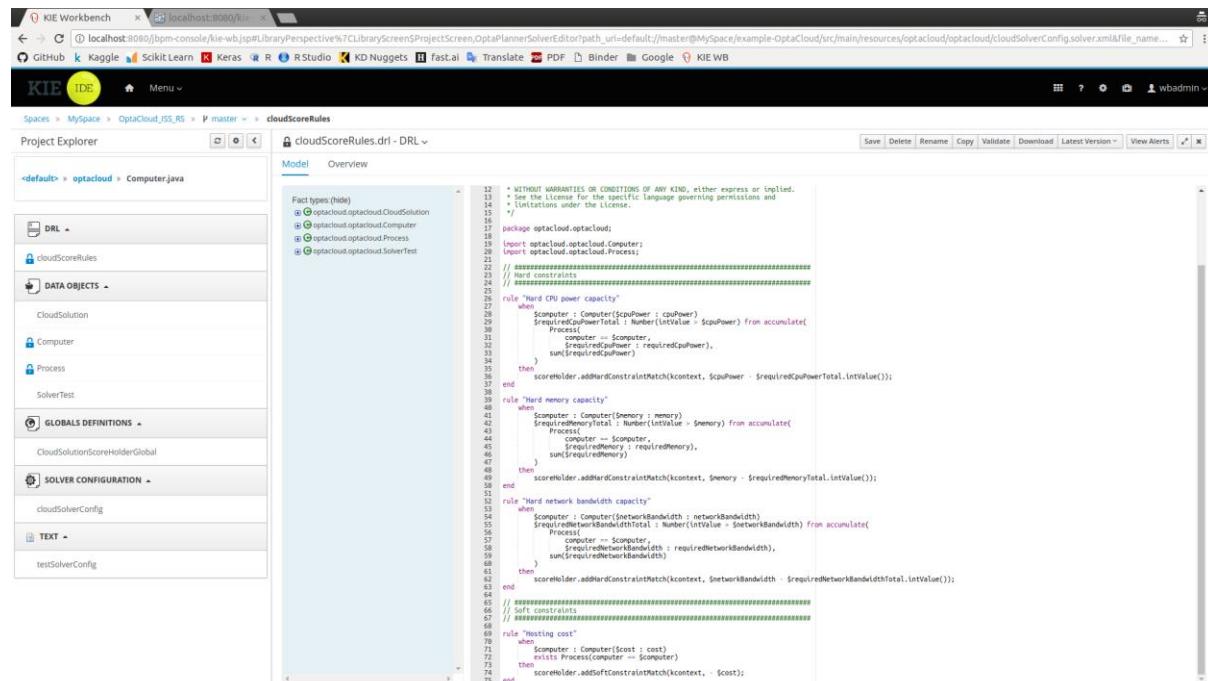
[Action] Add field: **id** (type: **long**) to Data Objects: **Computer**, **Process**, **CloudSolution**

```

1 /**
2  * Copyright 2012 Red Hat, Inc. and/or its affiliates.
3  * 
4  * Licensed under the Apache License, Version 2.0 (the "License");
5  * you may not use this file except in compliance with the License.
6  * You may obtain a copy of the License at
7  * 
8  *      http://www.apache.org/licenses/LICENSE-2.0
9  * 
10 * Unless required by applicable law or agreed to in writing, software
11 * distributed under the License is distributed on an "AS IS" BASIS,
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16 
17 package optacloud.optacloud;
18 
19 import java.util.ArrayList;
20 
21 import org.kie.api.KieContainer;
22 import org.kie.api.runtime.KieContainer;
23 import org.kie.api.runtime.KieServices;
24 import org.kie.api.runtime.KieServicesFactory;
25 import org.kie.api.runtime.KieContainerFactory;
26 import org.optaplanner.core.api.solver.SolverFactory;
27 import org.optaplanner.core.api.solver.Solver;
28 
29 public class SolverTest {
30 
31     @Test
32     public void solver() {
33         KieContainer kieContainer = KieServices.Factory.get().getKieContainer(KieContainerFactory.getClassLoader());
34         SolverFactory solverFactory = KieContainerFactory.createFromKieContainer(kieContainer);
35         solverFactory.setKieContainer(kieContainer);
36         solverFactory.setKieContainerResource("optacloud/testSolverConfig.xml");
37         SolverCloudSolution solver = solverFactory.buildSolver();
38         solver.solveGetSolution();
39     }
40 
41     private CloudSolution getSolution() {
42         Computer computer = new Computer(1000,
43                                         1000,
44                                         1000,
45                                         1000);
46         Process process = new Process(1000,
47                                      1000,
48                                      null);
49         return new CloudSolution(Arrays.asList(computer),
50                                Arrays.asList(process),
51                                null);
52     }
53 }
54 
55 */
56 
```

Above: unused **SolverTest** java program

[Action] To prevent Build/Deploy error, comment all code in Java script:
example-OptaCloud/src/test/java/optacloud/optacloud/SolverTest.java

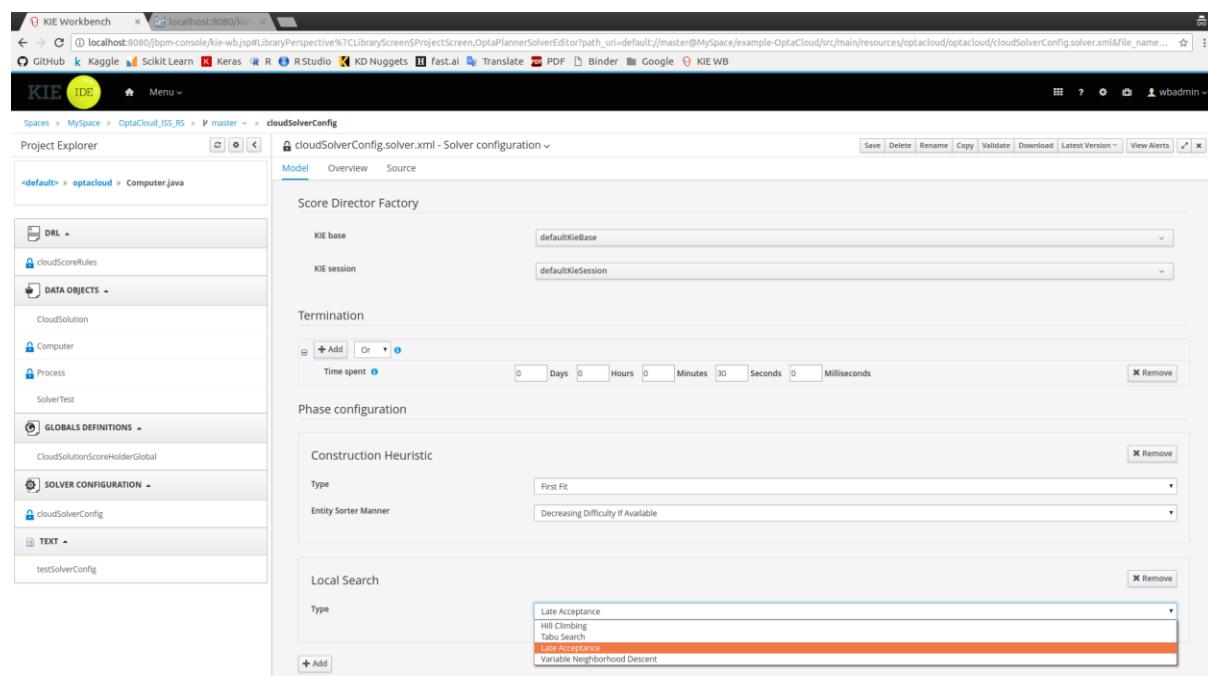


```

12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 * See the License for the specific language governing permissions and
14 * limitations under the License.
15 */
16 package optacloud.optacloud;
17
18 import optacloud.optacloud.Computer;
19 import optacloud.optacloud.Process;
20
21 //#####
22 // Hard constraints
23 //#####
24
25 rule "Hard CPU power capacity"
26 when
27   Computer : Computer($cpuPower : $cpuPower)
28   $requiredCpuPowerTotal : Number($intValue = $cpuPower) from accumulate(
29     Process
30     computer == Computer,
31     $cpuPower : requiredCpuPower,
32     sum($cpuPower)
33   )
34   then
35     scoreholder.addHardConstraintMatch($context, $cpuPower - $requiredCpuPowerTotal.intValue());
36 end
37
38 rule "Hard memory capacity"
39 when
40   Computer : Computer($memory : memory)
41   $requiredMemoryTotal : Number($intValue = $memory) from accumulate(
42     Process
43     computer == Computer,
44     $requiredMemory : requiredMemory,
45     sum($requiredMemory)
46   )
47   then
48     scoreholder.addHardConstraintMatch($context, $memory - $requiredMemoryTotal.intValue());
49 end
50
51 rule "Hard network bandwidth capacity"
52 when
53   Computer : Computer($networkBandwidth : networkBandwidth)
54   $requiredNetworkBandwidthTotal : Number($intValue = $networkBandwidth) from accumulate(
55     Process
56     computer == Computer,
57     $requiredNetworkBandwidth : requiredNetworkBandwidth,
58     sum($requiredNetworkBandwidth)
59   )
60   then
61     scoreholder.addHardConstraintMatch($context, $networkBandwidth - $requiredNetworkBandwidthTotal.intValue());
62 end
63
64 //#####
65 // Soft constraints
66 //#####
67
68 rule "hosting cost"
69 when
70   Computer : Computer($cost : cost)
71   exists Process(computer == Computer)
72   then
73     scoreholder.addSoftConstraintMatch($context, -$cost);
74 end
75

```

Above: constraints using Drools rule



Score Director Factory

- KIE base: defaultKieBase
- KIE session: defaultKieSession

Termination

- + Add: Time spent: 0 Days 0 Hours 0 Minutes 30 Seconds 0 Milliseconds

Phase configuration

Construction Heuristic

- Type: First Fit
- Entity Sorter Manner: Decreasing Difficulty If Available

Local Search

- Type: Late Acceptance, Hill Climbing, Tabu Search, Variable Neighborhood Descent

Above: solver configuration

The screenshot shows the KIE Workbench interface with the 'cloudSolverConfig.solver.xml' file open in the central editor. The XML content defines solver settings like time limits and search strategies.

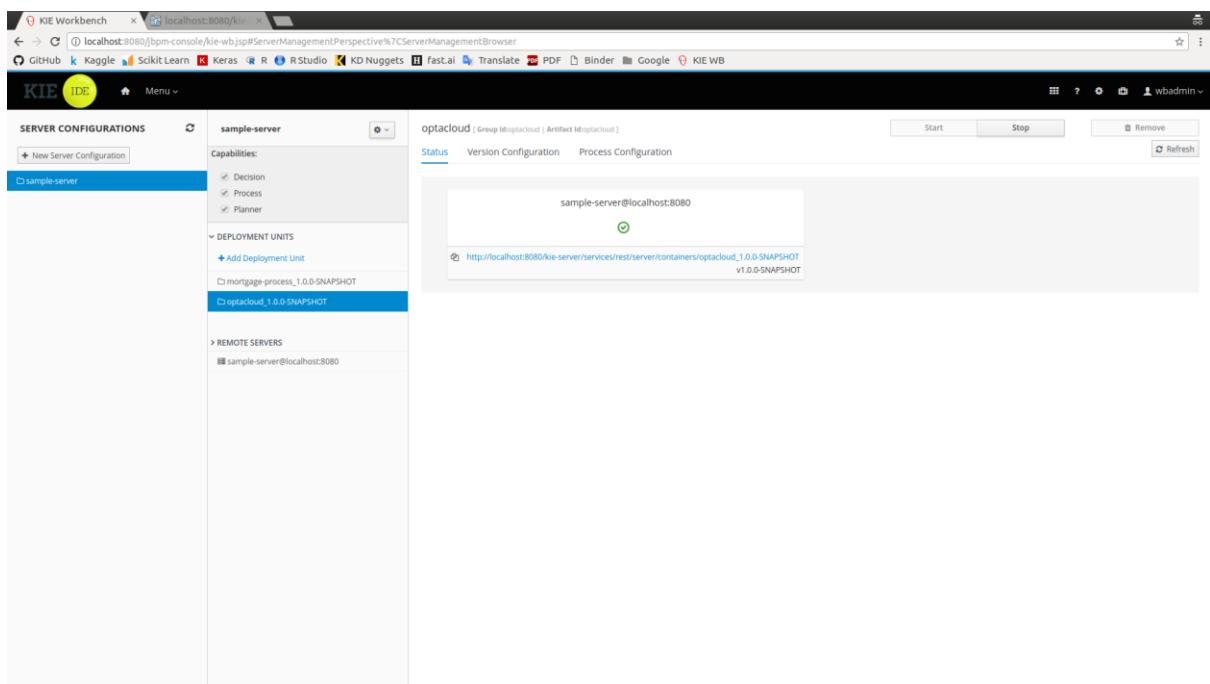
```

<?xml version="1.0" encoding="UTF-8"?>
<solver xstreamId="1">
    <constraintFactory xstreamId="2">
        <secondsSpentLimit>4</secondsSpentLimit>
        <millisecondsSpentLimit>400000000</millisecondsSpentLimit>
        <secondsSpentLimit>30</secondsSpentLimit>
        <millisecondsSpentLimit>30000000000</millisecondsSpentLimit>
        <hoursSpentLimit>4</hoursSpentLimit>
        <daysSpentLimit>1</daysSpentLimit>
    </termination>
    <constructionHeuristic xstreamId="3">
        <constructionHeuristicType>FIRST_FIT</constructionHeuristicType>
        <entitySorterName>DECREASING_DIFFICULTY_IF_AVAILABLE</entitySorterName>
    </constructionHeuristic>
    <localSearch xstreamId="4">
        <acceptanceCriterions>
            <acceptanceCriterion>OPEN_DATE_ACCEPTANCE</acceptanceCriterion>
        </acceptanceCriterions>
    </localSearch>
</solver>

```

Above: solver configuration (xml)

example-OptaCloud/src/main/resources/optacloud/optacloud/cloudSolverConfig.solver.xml

**Above: solver deployment to KIE Server**

```

<?xml version="1.0" encoding="UTF-8"?>
<response type="SUCCESS" asp="Info for container optacloud_1.0.0-SNAPSHOT">
  <<kie></kie>>
  <container container-alias="optacloud" container-id="optacloud_1.0.0-SNAPSHOT" status="STARTED">
    <config-items>
      <item>
        <itemName>KSession</itemName>
        <itemValue/>
        <itemType>BPM</itemType>
      </item>
      <item>
        <itemName>KSession</itemName>
        <itemValue/>
        <itemType>BPM</itemType>
      </item>
      <item>
        <itemName>MergeMode</itemName>
        <itemValue>MERGE COLLECTIONS</itemValue>
        <itemType>BPM</itemType>
      </item>
      <item>
        <itemName>RuntimeStrategy</itemName>
        <itemValue>SINGLETON</itemValue>
        <itemType>BPM</itemType>
      </item>
    </config-items>
    <messages>
      <message>
        Container optacloud_1.0.0-SNAPSHOT successfully created with module optacloud:optacloud:1.0.0-SNAPSHOT.
      </message>
    </messages>
    <resolved-ids>
      <resolved-id>
        <artifact-id>optacloud</artifact-id>
        <group-id>optacloud</group-id>
        <version>1.0.0-SNAPSHOT</version>
        <resolved-release-id>
          <artifact-id>optacloud</artifact-id>
          <group-id>optacloud</group-id>
          <version>1.0.0-SNAPSHOT</version>
        </resolved-release-id>
        <kie-container>
          <status>DISPOSED</status>
        </kie-container>
      </resolved-id>
    </resolved-ids>
  </container>
</response>

```

Above: solver web service end point

User: kieserver

Password: kieserver1!



PUT – Initiate a solver object-instance

PUT http://localhost:8080/kie-server/...
POST http://localhost:8080/kie-server/ser...
GET http://localhost:8080/kie-server/ser...

PUT http://localhost:8080/kie-server/services/rest/server/containers/employeeRostering_1.0.0-SNAPSHOT/solvers/cloudBalancingSolver

Headers (3)

Key	Value	Description
Authorization	Basic a2Iic2VylmVyOmtpZDNlcnZgEh	
X-KIE-Content-Type	xstream	
Content-Type	application/xml	

Body

```
<solver-instance>
<solver-config-file>optacloud/optacloud/cloudSolverConfig.solver.xml</solver-config-file>
</solver-instance>
```

Above: REST-API PUT via Postman tool

PUT http://localhost:8080/kie-server/...
POST http://localhost:8080/kie-server/ser...
GET http://localhost:8080/kie-server/ser...

PUT http://localhost:8080/kie-server/services/rest/server/containers/employeeRostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver

Body

```
<solver-instance>
<solver-config-file>optacloud/optaCloud/cloudSolverConfig.solver.xml</solver-config-file>
</solver-instance>
```

Above: REST-API PUT via Postman tool

```
<solver-instance>
<solver-config-file>optacloud/optacloud/cloudSolverConfig.solver.xml</solver-config-file>
</solver-instance>
```



POST – Start the solver with biz context

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. A collection named 'RestAPI optacloud' is selected, containing three requests: PUT, POST, and GET. The main workspace shows a POST request to `http://localhost:8080/kie-server/services/rest/server/containers/optacloud_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/state/solving`. The 'Headers' tab is active, listing:

KEY	VALUE	DESCRIPTION
Authorization	Basic a2lC2VydmlvY0mpZXNlczgjH	
X-KIE-ContentType	xstream	
Content-Type	application/xml	

The 'Body' tab shows an empty XML payload. The 'Test Results' tab displays the response headers and body, which include:

```

Expires - 0
Cache-Control - no-cache, no-store, must-revalidate
X-Powered-By - Undertow/1
Server - WildFly/11
Pragma - no-cache
Date - Wed, 13 Feb 2019 11:45:53 GMT
Connection - keep-alive
X-KIE-ConversationId - %27sample-server%27%3A%27optacloud_1.0.0-SNAPSHOT%27%3A%27optacloud%3Aoptacloud%3A1.0.0-SNAPSHOT%27%3A%27e11755-162b-4fe8-8a52-b5d368659e3%27
Content-Type - application/xml;charset=UTF-8
Content-Length - 0
  
```

Above: REST-API POST via Postman tool

```

1<?xml version="1.0" encoding="UTF-8"?>
2<aiplanning-problem class="optacloud_optacloud.CloudSolution" id="1">
3<!-->
4<!-->
5<!-->
6<!-->
7<!-->
8<!-->
9<!-->
10<!-->
11<!-->
12<!-->
13<!-->
14<!-->
15<!-->
16<!-->
17<!-->
18<!-->
19<!-->
20<!-->
21<!-->
22<!-->
23<!-->
24<!-->
25<!-->
26<!-->
27<!-->
28<!-->
29<!-->
30<!-->
31<!-->
32<!-->
33<!-->
34<!-->
35<!-->
36<!-->
37<!-->
38<!-->
39<!-->
40<!-->
41<!-->
42<!-->
43<!-->
44<!-->
45<!-->
46<!-->
47<!-->
48<!-->
49<!-->
50<!-->

```

Headers (10) Test Results

Expires - 0

Cache-Control - no-cache, no-store, must-revalidate

Status: 200 OK Time: 91 ms Size: 440 B Save

Above: REST-API POST via Postman tool

```
<planning-problem class="optacloud.optacloud.CloudSolution" id="1">
  <id>0</id>
  <computerList id="2">
    <optacloud.optacloud.Computer id="3">
      <id>0</id>
      <cpuPower>24</cpuPower>
      <memory>96</memory>
      <networkBandwidth>16</networkBandwidth>
      <cost>4800</cost>
    </optacloud.optacloud.Computer>
    <optacloud.optacloud.Computer id="4">
      <id>1</id>
      <cpuPower>6</cpuPower>
      <memory>4</memory>
      <networkBandwidth>6</networkBandwidth>
      <cost>660</cost>
    </optacloud.optacloud.Computer>
  </computerList>
  <processList id="5">
    <optacloud.optacloud.Process id="6">
      <id>0</id>
      <requiredCpuPower>1</requiredCpuPower>
      <requiredMemory>1</requiredMemory>
      <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
    <optacloud.optacloud.Process id="7">
      <id>1</id>
      <requiredCpuPower>3</requiredCpuPower>
      <requiredMemory>6</requiredMemory>
      <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
    <optacloud.optacloud.Process id="8">
      <id>2</id>
      <requiredCpuPower>1</requiredCpuPower>
      <requiredMemory>1</requiredMemory>
      <requiredNetworkBandwidth>3</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
    <optacloud.optacloud.Process id="9">
      <id>3</id>
      <requiredCpuPower>1</requiredCpuPower>
      <requiredMemory>2</requiredMemory>
      <requiredNetworkBandwidth>11</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
    <optacloud.optacloud.Process id="10">
      <id>4</id>
      <requiredCpuPower>1</requiredCpuPower>
      <requiredMemory>1</requiredMemory>
      <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
    <optacloud.optacloud.Process id="11">
      <id>5</id>
      <requiredCpuPower>1</requiredCpuPower>
      <requiredMemory>1</requiredMemory>
      <requiredNetworkBandwidth>5</requiredNetworkBandwidth>
    </optacloud.optacloud.Process>
  </processList>
</planning-problem>
```



GET – Obtain best solution (xml)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', there's a section for 'RestAPI optacloud' with three requests: PUT, POST, and GET. The main workspace shows a GET request to `http://localhost:8080/kie-server/services/rest/server/containers/employeerostering_1.0.0-SNAPSHOT/solvers/EmployeeRosteringSolver/bestsolution`. The 'Headers' tab is selected, displaying the following headers:

KEY	VALUE	DESCRIPTION
Key	Value	Description

The response body shows the XML output of the API call, which includes headers like Cache-Control, X-Powered-By, and Content-Type, and a large XML document representing the best solution.

Above: REST-API GET via Postman tool

The screenshot shows the Red Hat JBoss Fuse Workbench interface. The top navigation bar includes 'File', 'Import', 'Runner', 'My Workspace', 'Invite', and various system icons. The left sidebar has sections for 'History', 'Collections', 'Trash', and a 'RestAPI optcloud' entry with three requests listed. The main workspace displays a browser-like view of a REST API call:

- URL: `http://localhost:8080/kie-server/services/rest/server/containers/employeeRostering.1.0-5APSHOT/solvers/EmployeeRosteringSolver/bestsolution`
- Method: GET
- Headers: Authorization (set to Bearer token)
- Body (Raw XML):

```
<xml version="1.0" encoding="UTF-8" standalone="yes">
<serverInstances>
  <container id="optcloud_1" isSnapshot="true" containerId="1">
    <solver id="cloudBalancingSolver" solverId="1" />
    <solverConfig file="optcloud/cloudServerConfig.xml" solver="xml"/>
    <score scoreClass="org.optcloud.core.api.score.buildin.hardsoft.HardSoftScore">hardf/546soft/score</score>
    <solverType>cloudServer</solverType>
    <networkBandwidth>http://www.w3.org/2001/XMLSchema-instance</networkBandwidth>
    <computerList>
      <cost>400</cost>
      <cpuPower>4</cpuPower>
      <id>1</id>
      <memory>4</memory>
      <networkBandwidth>100</networkBandwidth>
    </computerList>
    <cost>400</cost>
    <cpuPower>4</cpuPower>
    <id>2</id>
    <memory>4</memory>
    <networkBandwidth>100</networkBandwidth>
  </container>
</serverInstances>
```
- Tests tab is selected.
- Status: 200 OK, Time: 26 ms, Size: 4.22 KB
- Buttons: Send, Save, Examples (dropdown), Cookies (dropdown), Bulk Edit, Learn, and several icons for file operations.

Above: REST-API GET via Postman tool

Above: REST-API GET via Postman tool

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
</solver-instance>
<container-id>optacloud_1.0.0-SNAPSHOT</container-id>
<solver-id>cloudBalancingSolver</solver-id>
<solver-config-file>optacloud/optacloud/cloudSolverConfig.solver.xml</solver-config-file>
<status>NOT SOLVING</status>
<score scoreClass="org.optaplanner.core.api.score.buildin.hardsoft.HardSoftScore">0hard/-5460soft</score>
<best-solution xsi:type="cloudSolution" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <computerList>
        <cost>4800</cost>
        <cpuPower>24</cpuPower>
        <id>0</id>
        <memory>96</memory>
        <networkBandwidth>16</networkBandwidth>
    </computerList>
    <computerList>
        <cost>660</cost>
        <cpuPower>6</cpuPower>
        <id>1</id>
        <memory>4</memory>
        <networkBandwidth>6</networkBandwidth>
    </computerList>
    <processList>
        <computer>
            <cost>660</cost>
            <cpuPower>6</cpuPower>
            <id>1</id>
            <memory>4</memory>
            <networkBandwidth>6</networkBandwidth>
        </computer>
        <id>0</id>
        <requiredCpuPower>1</requiredCpuPower>
        <requiredMemory>1</requiredMemory>
        <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </processList>
    <processList>
        <computer>
            <cost>4800</cost>
            <cpuPower>24</cpuPower>
            <id>0</id>
            <memory>96</memory>
            <networkBandwidth>16</networkBandwidth>
        </computer>
        <id>1</id>
        <requiredCpuPower>3</requiredCpuPower>
        <requiredMemory>6</requiredMemory>
        <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </processList>
    <processList>
        <computer>
            <cost>4800</cost>
            <cpuPower>24</cpuPower>
            <id>0</id>
            <memory>96</memory>
            <networkBandwidth>16</networkBandwidth>
        </computer>
        <id>2</id>
        <requiredCpuPower>1</requiredCpuPower>
        <requiredMemory>1</requiredMemory>
        <requiredNetworkBandwidth>3</requiredNetworkBandwidth>
    </processList>
    <processList>
        <computer>
            <cost>4800</cost>
            <cpuPower>24</cpuPower>
            <id>0</id>
            <memory>96</memory>
            <networkBandwidth>16</networkBandwidth>
        </computer>
        <id>3</id>
        <requiredCpuPower>1</requiredCpuPower>
        <requiredMemory>2</requiredMemory>
        <requiredNetworkBandwidth>11</requiredNetworkBandwidth>
    </processList>
    <processList>
        <computer>
            <cost>4800</cost>
            <cpuPower>24</cpuPower>
            <id>0</id>
            <memory>96</memory>
            <networkBandwidth>16</networkBandwidth>
        </computer>
        <id>4</id>
        <requiredCpuPower>1</requiredCpuPower>
        <requiredMemory>1</requiredMemory>
        <requiredNetworkBandwidth>1</requiredNetworkBandwidth>
    </processList>
    <processList>
        <computer>
            <cost>660</cost>
            <cpuPower>6</cpuPower>
            <id>1</id>
            <memory>4</memory>
            <networkBandwidth>6</networkBandwidth>
        </computer>
        <id>5</id>
        <requiredCpuPower>1</requiredCpuPower>
        <requiredMemory>1</requiredMemory>
        <requiredNetworkBandwidth>5</requiredNetworkBandwidth>
    </processList>
    <score>0hard/-5460soft</score>
    <id>0</id>
</best-solution>
</solver-instance>

```

Reference

- https://docs.optaplanner.org/7.12.0.Final/optaplanner-wb-es-docs/html_single/
- <http://www.optaplanner.org/learn/useCases/cloudOptimization.html>
- <http://www.optaplanner.org/learn/slides/optaplanner-presentation/training.html#/4/26>
- https://docs.jboss.org/optaplanner/release/latestFinal/optaplanner-wb-es-docs/html_single/

Workshop 2.2 [Individual]

Choose one use case; propose relevant enhancements; then import, analyze, adapt, enhance, and solve/execute.

Enhancement example 1: [Reuse KIE Workbench OptaCloud REST-API project]

Enrich the Cloud Balancing domain model and add extra constraints such as:

- Some Process running deep learning neural network models can require graphical processing units GPU chips, so these processes should (or must) be assigned to computers with sufficient number of GPU chips.
- Each Process belongs to a Service. A computer might crash, so processes running the same service must be assigned to different computers.
- Each Computer is located in a Building. A building might burn down, so processes of the same services should (or must) be assigned to computers in different buildings.

Reference https://docs.optaplanner.org/latest/optaplanner-docs/html_single/index.html#cloudBalancingBeyondThisTutorial

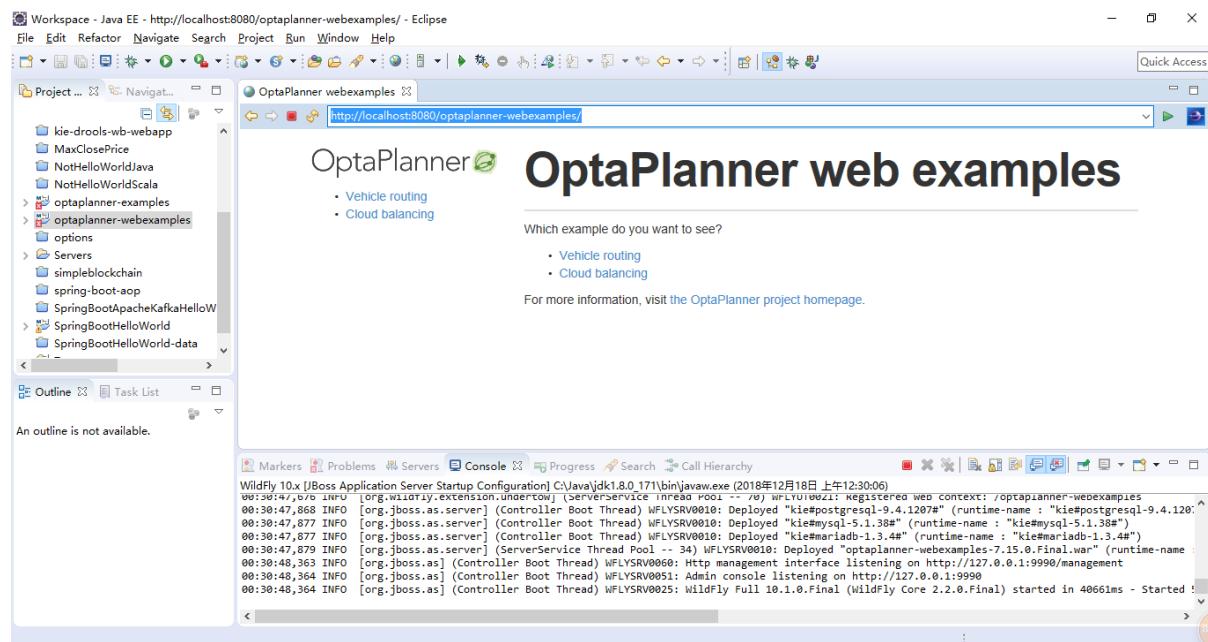
Enhancement example 2: [Reuse any KIE Workbench project or Eclipse project]

Propose and implement/enhance your own relevant business use case.

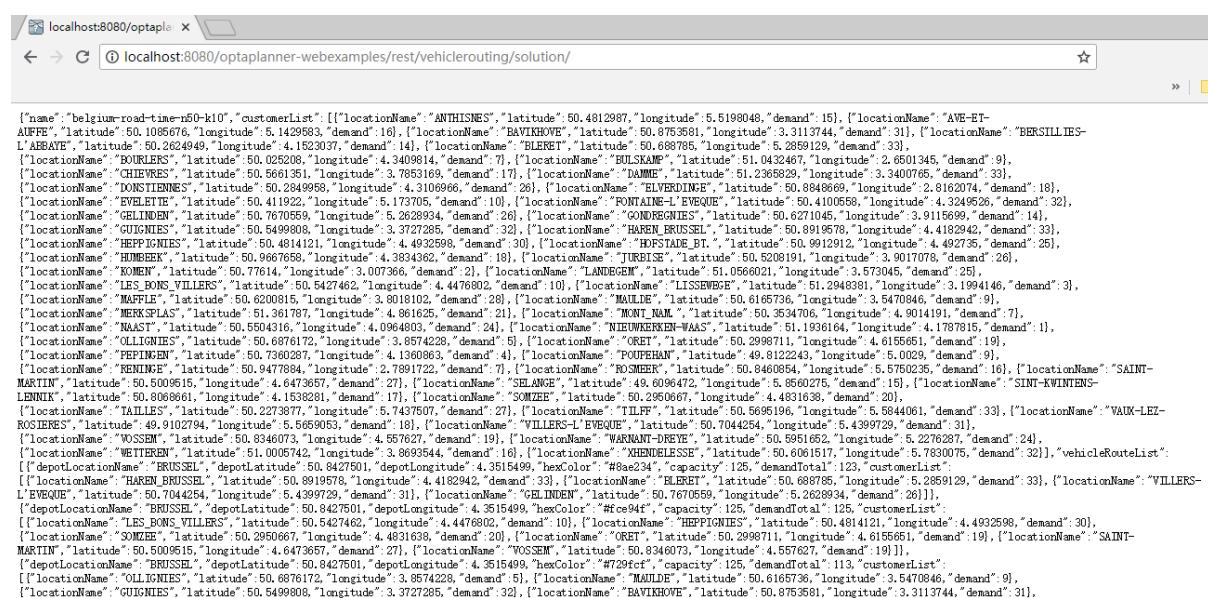
Enhancement example 3: [Reuse KIE Workbench OptaCloud REST-API project]

Enhance the Cloud Balancing system by introducing a comprehensive user interface for end user to

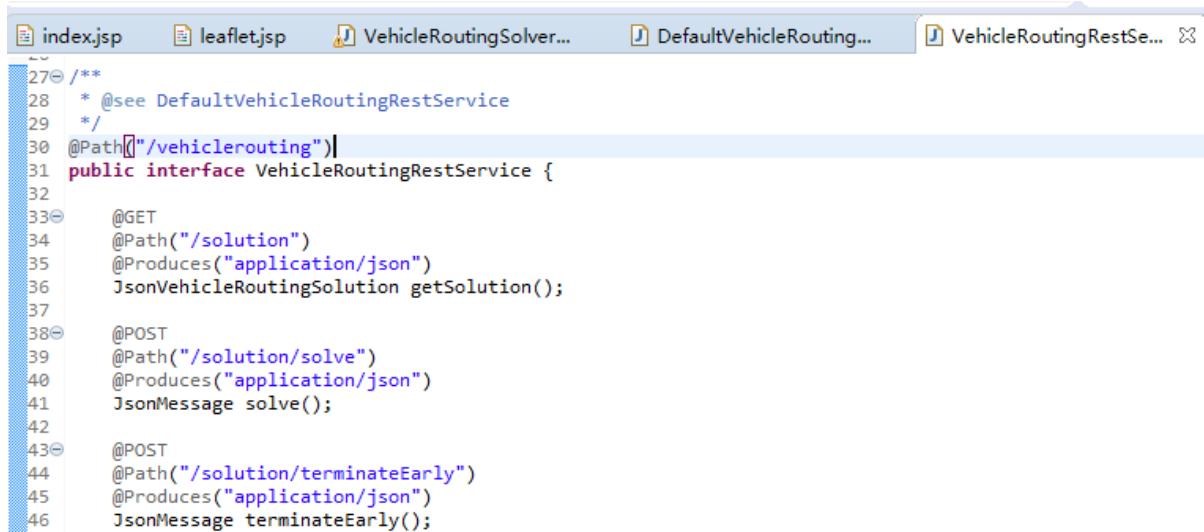
1. [Input] Key in problem configuration:
 - a. Computer instances with: cpuPower, memory, networkBandwidth, cost.
 - b. Process instances with: requiredCpuPower, requiredMemory, and requiredNetworkBandwidth.
2. [Output] Display graphical final solution of cloud computer assignments:
 - a. **Use REST-API for (web based) UI to communicate with KIE web server's solver engine.**
 - b. Hint: OptaPlanner web example application as reference:
<https://github.com/kiegroup/optaplanner/tree/master/optaplanner-webexamples>



Above: example web applications



Above: example solution enquiry via RESTful API

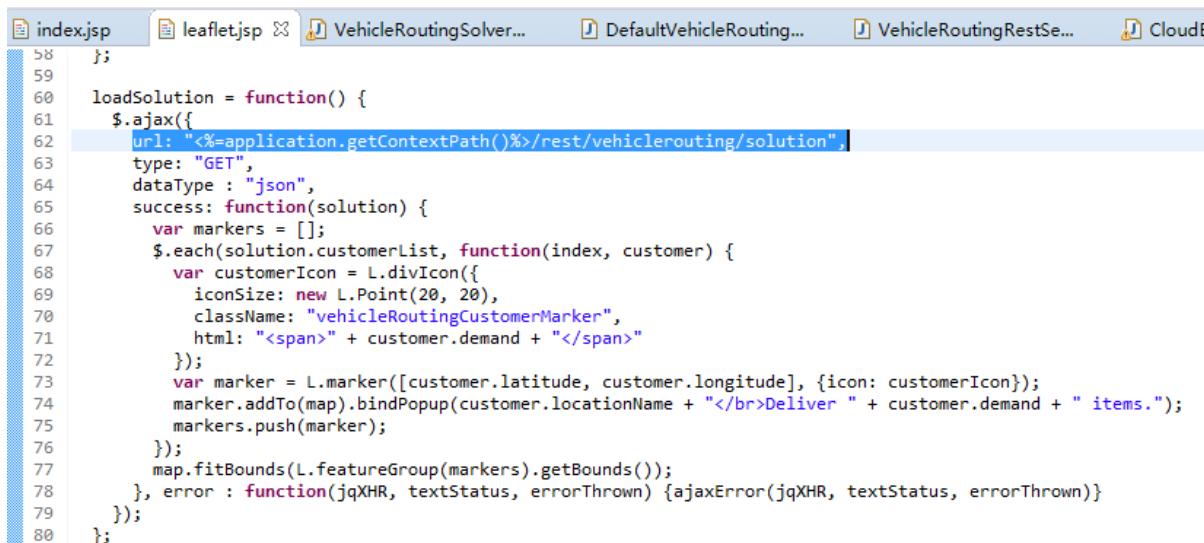


```

27 /**
28  * @see DefaultVehicleRoutingRestService
29 */
30 @Path("/vehiclerouting")
31 public interface VehicleRoutingRestService {
32
33     @GET
34     @Path("/solution")
35     @Produces("application/json")
36     JsonVehicleRoutingSolution getSolution();
37
38     @POST
39     @Path("/solution/solve")
40     @Produces("application/json")
41     JsonMessage solve();
42
43     @POST
44     @Path("/solution/terminateEarly")
45     @Produces("application/json")
46     JsonMessage terminateEarly();

```

Above: example KIE server interaction via RESTful API



```

58 };
59
60 loadSolution = function() {
61     $.ajax({
62         url: "<%=application.getContextPath()%>/rest/vehiclerouting/solution",
63         type: "GET",
64         dataType : "json",
65         success: function(solution) {
66             var markers = [];
67             $.each(solution.customerList, function(index, customer) {
68                 var customerIcon = L.divIcon({
69                     iconSize: new L.Point(20, 20),
70                     className: "vehicleRoutingCustomerMarker",
71                     html: "<span>" + customer.demand + "</span>"
72                 });
73                 var marker = L.marker([customer.latitude, customer.longitude], {icon: customerIcon});
74                 marker.addTo(map).bindPopup(customer.locationName + "<br>Deliver " + customer.demand + " items.");
75                 markers.push(marker);
76             });
77             map.fitBounds(L.featureGroup(markers).getBounds());
78         }, error : function(jqXHR, textStatus, errorThrown) {ajaxError(jqXHR, textStatus, errorThrown)}
79     });
80 };

```

Above: example server end point

Reference

- <https://github.com/kiegroup/optaplanner/tree/master/optaplanner-webexamples>
- https://docs.optaplanner.org/latest/optaplanner-docs/html_single/index.html#quickStart

The End of Workshop Project Guide