Melodic
Big data cloud

Title:

Integration release and initial test environment

Abstract

A reliable multi-cloud project should provide to the customer a high-quality platform that allows to run data-intensive applications in a geographically distributed infrastructure; it should ensure the security and privacy of data, enable transparent deployment of applications on the multi-cloud infrastructure and minimise the risk of failure in a live environment. These very requirements are met within the Melodic project.

The Melodic platform will enable data-intensive applications to run within defined security, cost, and performance boundaries seamlessly on geographically distributed and federated cloud infrastructures. Melodic will thereby realise the potential of heterogeneous cloud environments for big data and data-intensive applications by transparently taking advantage of distinct characteristics of available private and public clouds, dynamically optimise resource utilisation, consider data locality, conform to the user's privacy needs and service requirements, and counter vendor lock-in. This document presents the two integration releases of Melodic (release 1.0 and release 1.5) and defines the corresponding testing components including the testing environments.

Multi-cloud Execution-ware for Large-scale Optimized Data-Intensive Computing

H2020-ICT-2016-2017
Leadership in Enabling and Industrial Technologies; Information and Communication Technologies

Grant Agreement No.:
731664

Duration:
1 December 2016
30 November 2019

www.melodic.cloud

Deliverable reference:
5.07

Date:
31 May 2018

Responsible partner:
7bulls

Editor(s):
Edyta Bańkowska

Author(s)
Edyta Bańkowska

Approved by:
Ernst Gunnar Gran

ISBN number:
N/A

Document URL:
http://www.melodic.cloud/deliverables/D5.07 Integration release and initial test environment.pdf

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Document	
Period Covered	M6-18
Deliverable No.	D5.07
Deliverable Title	Integration release and initial test environment
Editor(s)	Edyta Bańkowska
Author(s)	Edyta Bańkowska
Reviewer(s)	Kyriakos Kritikos, Feroz Zahid, Tomasz Przeździęk
Work Package No.	5
Work Package Title	Integration and security
Lead Beneficiary	7bulls
Distribution	PU
Version	1.0
Draft/Final	Final
Total No. of Pages	56

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Index of Tables

Index of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

1 Introduction

This document presents information about the two integration releases of the Melodic project, initial testing environments, processes of testing, reported bugs and created test cases. This work has been done during the first sixteen months of the project and briefly shows the steps of the test process of the project. It covers release 1.0 delivered as planned at the end of the first year of the project and release 1.5 delivered at the end of the sixteenth month of the project.

The test cases include initial deployment testing, global reconfiguration and local reconfiguration testing, metric management testing, and reasoning related testing. Apart from functional requirements, also non-functional testing requirements are covered in order to verify the proper implementation of the non-functional features of the Melodic system. The non-functional testing test cases include testing of fault handling, performance testing, security testing, and other non-functional testing.

The document starts with an overview of the Melodic software components. Next, the configurations of the environments are described, followed by an installation guide, a testing guide and a description of executed Test Cases. The last main chapter relates to the reported bugs.

1.1 Integration releases

For the Melodic project, as presented in Description of Action, three releases was planned:

- Release 1 – planned for 30 November 2017
- Release 2 – planned for 30 November 2018
- Release 3 – planned for 30 November 2019

Based on a requirement review and scope planning during the Melodic Plenary Meetings in Oslo 07.06.2017 and in Warsaw 18.09.2017, it was decided to add a supplementary integration release, called release 1.5. The first integration release, as presented in the Description of Action should only integrate underlying frameworks without changing their features. To make it possible to fully evaluate use case applications after the first release, the additional release 1.5 was introduced with a set of new features and improvements.

The scope of release 1.0 is as follows:

- Integration of the selected components from the underlying frameworks
- New integration layer using Enterprise Service Bus (ESB) and Business Process Management (BPM) orchestration
- Docker containers introduction and microservice architecture for Melodic
- Rewriting the Adapter component and improving the CP Generator component

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

The scope of release 1.5 is as follows:

- Introduction of a new approach to optimization of the cloud deployment, limiting the possible solution space to speed up solution finding
- A new, more flexible and advanced monitoring and reporting solution based on Esper (instead of the Metric Collector from the PaaSage[1] framework).
- All components of the Melodic platform implemented in Docker images and run within Docker Swarm to allow for very easy and flexible deployment of the platform itself
- Advanced utility functions for use case applications designed and implemented

To achieve high quality of the software products, ensure security and privacy, enable transparent deployment of software and minimise the risk of failure in the real world, appropriate software testing tools and techniques will be used for both releases.

1.2 Structure of the document

This deliverable describes the two integration releases and their initial test environments, and contains the following information which, structured in the subsequent chapters:

- Chapter 2, 'Components – The Melodic Architecture': A brief summary of the Melodic architecture along with the release 1.0 and 1.5 components
- Chapter 3, 'Testing environments': Information about the environments currently used in the project
- Chapter 4, 'The Melodic platform installation guide': Requirements and steps on how to install the Melodic platform
- Chapter 5, 'Testing Guide': Detailed instructions on how to execute Test Cases;
- Chapter 6, 'Test Cases': A comparison of all executed Test Cases during release 1.0 and 1.5, as further detailed in Appendix A, 'Test Cases Release 1.0 and Release 1.5'
- Chapter 7, 'Reported bugs': Information about all bugs which have been detected during the testing of release 1.0 and release 1.5

The deliverable ends with a short summary (Chapter 8).

[1] https://www.paasage.eu/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2 Components – The Melodic Architecture

Figure 1 presents an overview of the Melodic architecture. The figure also covers the high-level interaction between the Melodic components, while a detailed architecture is presented in "D2.2 Architecture and initial feature definitions" [1]. The main Melodic sub-systems are:

- **Upperware**: Applications and data models created through the modelling interfaces, in the form of CAMEL, are given as input to the Melodic Upperware. The job of the Upperware is to calculate optimal data placements and application deployments on dynamically acquired Cross-Cloud resources in accordance with the specified application and data models in CAMEL, as well as in consideration of the current Cloud performance, workload situation, and costs. An overview of the Upperware Components is shown in Figure 2, further detailed in [1].
- **Executionware**: The Executionware is responsible for the actual deployment of the Cloud application and its monitoring infrastructure, as well as the corresponding publishing of measurement information to the Upperware.
- **Integration layer**: The components in the Melodic platform are integrated through two separate integration layers, the Control Plane and the Monitoring Plane (blue boxes in Figure 1), each bringing its own set of unique requirements.

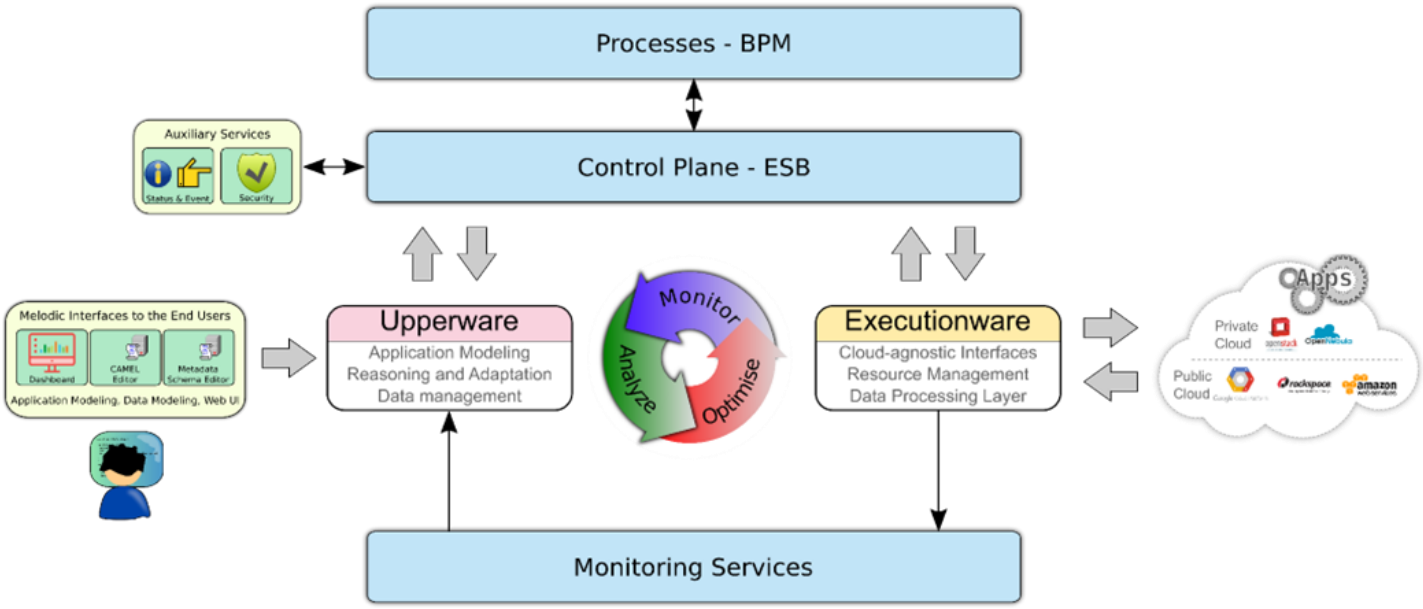


Figure 1: Overview of the Melodic architecture

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

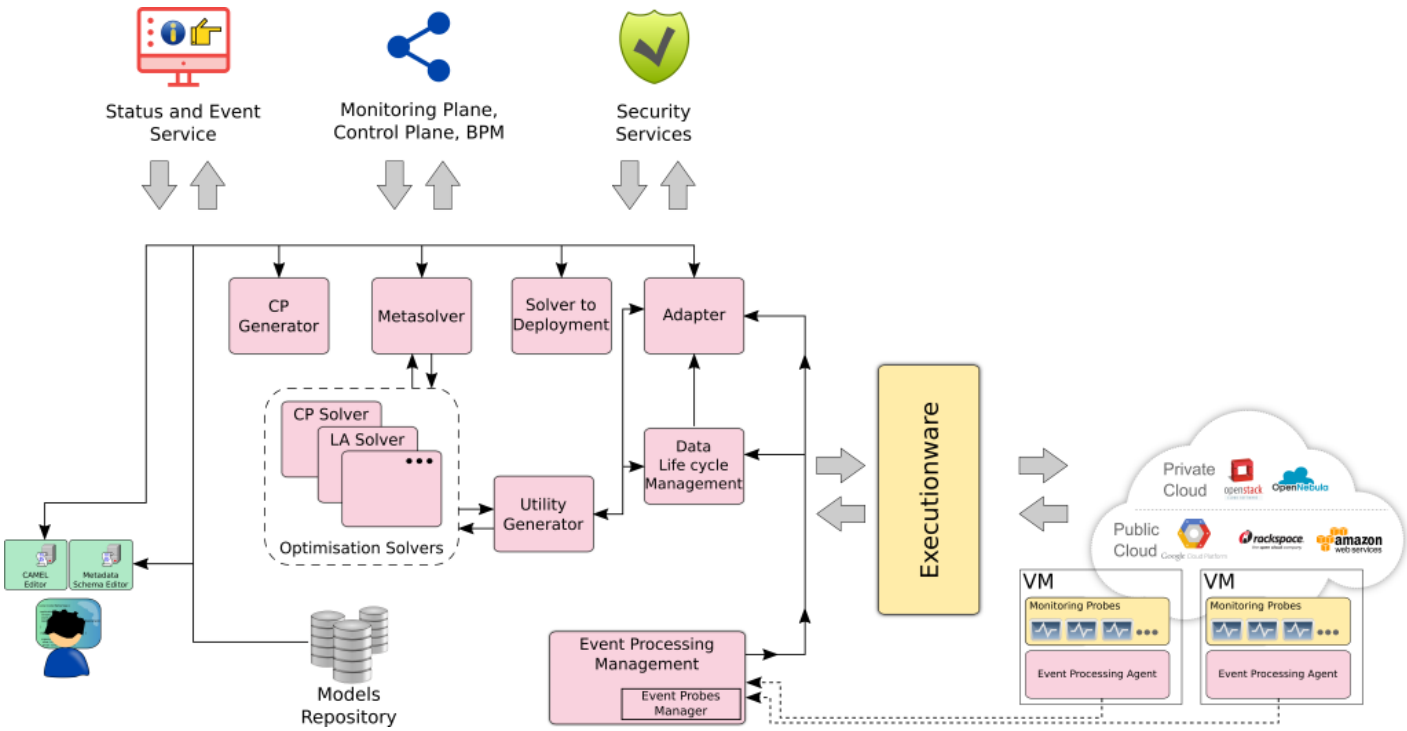


Figure 2: Overview of the Upperware Components

2.1 Software Components

Table 1 lists central components in Melodic for release 1.0 and 1.5, including their key features and the sub-systems they belong to. D2.2 provides further details [1].

Table 1: Main components used in the Melodic project, with particular relevance for testing

Component	Description, key feature	Sub-system	Framework
CP Generator	Profiling of the application and preparation of the constraint programming (CP) model	Upperware	PaaSage
CP Solver	Solving all types of problems encoded in the CP model using gradient descent approach.	Upperware	PaaSage
Solver-To-Deployment	Transforming CP models encompassing the solution produced by solvers to a provider-specific deployment model.	Upperware	PaaSage
Adapter	Deployment control and adaptation of multi-cloud applications	Upperware	PaaSage

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Cloudiator[2]	Cloudiator (server part)	Deploying an application and infrastructure to the Cloud Providers.	Executionware	Cloudiator is the result of the PaaSage project and was extended in the Cactos project.
	Cloudiator (VM part)	Components deployed on the created Virtual Machines (VMs)		
Camunda[3] with status/event service	Camunda is an open-source workflow engine written in Java that can execute business processes		Integration layer	Melodic
ESB	Ensure communication between all components (The exception is CDO [1])		Integration layer	Melodic
ESPER	Gathering metrics		Upperware	Melodic

3 Testing environments

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, the environment supports test execution with hardware, software and network configured. Such environments may vary significantly in size: the development environment is typically an individual developer's workstation, while the production environment may be a network of many geographically distributed machines in data centres, or virtual machines in cloud computing. The code, data, and configuration may be deployed in parallel, and need not be connected to the corresponding tier; For example, pre-production code may connect to a production database.

We used two different testing environments for release 1.0 at 7bulls and Ulm, respectively. In addition, for release 1.5 we used an environment at Simula. The corresponding parameters of the testing environments are provided in Section 3.1 to Section 3.3. Non-functional and functional testing were executed on the same, common set of environments at 7bulls, Ulm or Simula.

3.1 Environment at Ulm

For Melodic release 1.0, we used two virtual machines (VMs) for testing on which we installed the Melodic platform, using the OpenStack-based Ulm datacentre "Omistack". The specification for the Ulm environment is provided in Table 2.

[2] http://cloudiator.org/

[3] https://camunda.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 2: Specification of the Ulm environment

Resource	Details
CPU	16 cores
RAM	128 GB
Storage	200 GB SSD, 400 GB HDD
OpenStack version	Ocata
Virtualization	KVM (Xen and Docker hosts are planned)
Host OS	CentOS 7
Network interfaces	10GbE

3.2 Environment in 7bulls

Similarly as for the Ulm environment, two VMs were created on Aruba at 7bulls for release 1.0 testing. The specification of the 7bulls environment is provided in

Table 3.

Table 3: Specification of the 7bulls environment

Resource	Details
CPU	X86_64/20 cores
RAM	40 GB
Storage	800 GB HDD
Virtualization	VMware
Host OS	Linux 14.10 for chef/16.04 for Docker
Network interfaces	1 GB/s

3.3 Environment in Simula

During release 1.5, all tests were executed on the Simula environment. For the Simula environment, four VMs were created, as further detailed in Table 4.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 4: Specification of the Simula environment

Resource	Details
CPU	Intel E5-2630 v4 @ 2.20GHz / 2 / 20, 40 threads in total
RAM	128 GB
Storage	186.3 GB SSD, 1.8 TB HDD
Virtualization	KVM / Linux kernel v4.10
Host OS	Ubuntu Linux 16.04 LTS (amd64)
Network interfaces	2 GB/s

4 The Melodic platform installation guide

Melodic, by improving performance, security, and scalability of Cloud applications, and in particular the data-intensive ones, will be directly beneficial to the Cloud application users. Melodic's multi- and Cross-Cloud application deployment features, will make it easier for the application developers to develop Cloud applications that can be tailored to the Cloud application user's preferences, be privacy-aware, and offer predictive services under a dynamically changing context and user load. In addition, as Melodic promises to improve cost-effectiveness of the Cloud applications by leveraging the economically best available Cloud offers from a variety of Cloud providers, low-cost Cloud services will in turn be offered to the end users due to open competition. Finally, Melodic will also foster innovation among Cloud providers and application developers, as once the vendor lock-in is overcome, the competitive advantages of the big providers should diminish. In this way, only the most competitive services in the market will prevail, which will be most beneficial to the Cloud users. This section describes how to install the Melodic platform from scratch.

4.1 Requirements for Melodic's machine

Table 5 details the hardware and operating system requirements for the current Melodic software, while Table 6 specifies the port numbers being used by Melodic.

Table 5: requirements for Melodic's machine

OS	Memory	Storage
Ubuntu 16.04	RAM: 16GB+	20GB+

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 6: open port numbers required by Melodic

Port Number	Protocol	Component	Purpose
22	TCP	SSH	Console
8080-8099	TCP	Components	REST endpoints of Melodic components (CP Generator, CP Solver, Solver-to-Deployment, Adapter, ESB)
9001-10000	TCP		
2036, 3306	TCP	CDO	MySQL database
80	TCP	UI	Cloudiator's web interface
4001	TCP	Lance	ETCD registry
9000	TCP	Colosseum	Cloudiator's REST API
8080	TCP	Axe	Time-series database
33034	TCP	Lance	RMI registry

4.2 Installation steps

- Login to the created virtual machine, for example using: ssh *username*@<*VM's IP*>

- Run the following command to download Melodic from the git repository (installation files):

```
git clone https://bitbucket.7bulls.eu/scm/mel/utils.git
```

- Edit Melodic installation script by using a text editor like vi:

```
vi ~/utils/melodic_installation/installMelodic.sh
```

- Configure the NODEGROUP value, for example using your own login:

```
NODEGROUP=jdoe
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Run the Melodic installation script (this installs Melodic and Cloudiator on the same VM):

```
~/utils/melodic_installation/installMelodic.sh install
```

- After installation, a new ".profile" is created in the home directory of the user. Load it by executing the following commands:

```
cd ~/
. .profile
```

- Verify that the Melodic configuration files contain the proper IP addresses (check files under ~/conf). In particular, if you are running Cloudiator on a different host you need to update Cloudiator's IP address in the *eu.melodic.integration.mule.properties* file. Make sure that the Cloudiator IP address is also correctly changed in the following files:

```
eu.melodic.integration.bpm.properties
eu.melodic.integration.mule.properties
eu.melodic.upperware.adapter.properties
eu.melodic.upperware.cpSolver.properties
eu.melodic.upperware.generator.properties
eu.melodic.upperware.solverToDeployment.properties
```

Example:

Updating Cloudiator IP (example: *eu.melodic.integration.mule.properties* file):

```
adapter.http.host=5.249.145.169
adapter.http.port=8097

#we change Cloudiator IP in this section
cloudiator.http.host=5.249.145.169
cloudiator.http.port=9000

cloudiator_proxy.http.host=0.0.0.0
cloudiator_proxy.http.port=8089
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Add Cloud Provider credentials to allow Melodic to download Provider Offers in the adapter configuration file. For instance, if you want Melodic to manage amazon machines, you need to edit file:

```
~/conf/eu.melodic.upperware.adapter.properties
```

Provide values to the following attributes:

```
clouds.endpoints.ec2=
clouds.logins.ec2=
clouds.passwords.ec2=
```

- Add API key to the generator configuration file (an API key allows access to Cloudiator's API). It can be received from the Cloudiator installation.

For example: cloudiatorV2.apiKey=abcdefghijk12345687

Now the machine is ready to download and run the latest Docker images from the Melodic artefact repository. To download and start the components, simply use the following command:

```
ddeploy
```

Running this for the first time can take some time as Docker Swarm is being initialised. After the above command, components should be started. You can check the status by running the following two commands:

```
dps
mping
```

The screenshot below shows a part of the output after executing the *dps* and *mping* commands:

```
ubuntu@melodicinstallation:~$ dps
sudo: unable to resolve host melodicinstallation
CONTAINER ID        IMAGE                                       COMMAND
d208f35a2a3d        88.99.85.63:5000/melodic/adapter:latest     "java -Djava.secur..."
1cef4e7ce5ce        88.99.85.63:5000/melodic/generator:latest   "java -Duser.timez..."
f45500838779        88.99.85.63:5000/melodic/cpsolver:latest    "java -Djava.secur..."
f92ef764f463        88.99.85.63:5000/melodic/cdoserver:latest   "/docker-entrypoin..."
979bdad3c361        88.99.85.63:5000/melodic/s2d:latest         "java -Djava.secur..."
62ef820bc2eb        88.99.85.63:5000/melodic/mule:latest        "/opt/mule/bin/mul..."
d75097055dd7        88.99.85.63:5000/melodic/process:latest     "java -Duser.timez..."
ubuntu@melodicinstallation:~$ mping
mule: 8088 status:  OK
mule: 8089 status:  OK
process: 8095 status:  OK
adapter: 8097 status:  OK
solver2deployment: 8096 status:  OK
cdoserver: 2036 status:  OK
generator: 8091 status:  OK
cpsolver: 8093 status:  OK
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

The process GUI should now be available at:

```
http://{PUBLIC_MELODIC_IP}:8095      (admin:admin)
```

The status of the Colosseum (Cloudiator) service can be checked with:

```
sudo service colosseum status
```

The log of Cloudiator can be found at:

```
/var/log/colosseum.log
```

The Cloudiator GUI should be available at:

```
http://{PUBLIC_CLOUDIATOR_IP}/ui   (john.doe@example.com : admin:admin)
```

The script to restart Coudiator with full wipe out of Colosseum data is available under:

```
~/cloudiator_reset.sh
```

4.3 Useful aliases

Below you find some useful commands to manage Melodic components.

Commands:

dps – displays running Docker containers (alias for *sudo Docker images*)

mping – tests connection to each of the components

drestart – stops and then starts all of the Melodic's components

dundeploy – stops all components

ddeploy – starts all components

~/logs$ tail -300f <component_name>.log – displays the log of the selected component

sudo docker stop <component_ID> – stops selected component

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5 Testing Guide

This section refers to the procedures, and in particular provides a manual, for performing all activities within the testing processes. Each Test Case has attached CAMEL or CP models. Every created and included model specifies either the testing application, including requirements, topology, metrics and credentials in case of CAMEL models, or the constraint optimisation problem to be solved during deployment reasoning in case of CP models.

5.1 How to execute Test Cases with attached CAMEL Models

This section presents the Quality Assurance guide of testing Test Cases with attached CAMEL models.

1. Login to the machine with the Melodic platform installed
2. Download the jar file from:
   https://s3.console.aws.amazon.com/s3/object/melodic.testing.data/cdo-uploader-1.0.1SNAPSHOT-jar-with-dependencies.jar
   Download *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar*
   a. In "/home/user/", create the hidden directory ".paasage" and put into this directory the configuration file for CDO named *eu.paasage.mddb.cdo.client.properties* which can be downloaded from:
   https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/?region=us-east-1&tab=overview
   b. In the *eu.paasage.mddb.cdo.client.properties* configuration file, change the IP address with the one of your virtual machine for the "host" property
   c. In the terminal use the command:
   java -jar cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar
3. Create the directory "models" in "/home/user/"
4. From https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/ download the CAMEL provider model files *AmazonEC2.xmi* and *OpenStackUlm.xmi*
5. Navigate to the "/home/user/models" directory and create the following sub-directory structure: *"upperware-models/fms"*
6. Move the downloaded CAMEL providers models *AmazonEC2.xmi* and *OpenStackUlm.xmi* to the "/home/user/models/upperware-models/fms" directory
7. Download the respective CAMEL model needed for the test case from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data in Melodic's JIRA (required CAMEL Model is always attached in Test Case)
8. Move the downloaded CAMEL model (having a *.xmi* postfix) into the "/home/user/models" directory

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

9. By using the tools **SoapUI** or **Postman** execute the following steps:
   a. Create new REST project with URL
      http://<VM's IP>:8088/api/frontend/deploymentProcess and body:

Sample body

```
# In "CamelModelName" user can use: "OneComponentApp" or "TwoComponentApp"
# Rest of the body doesn't need to be changed

{     "applicationId": "CamelModelName",          "watermark": {
            "user": "User-Test",
            "system": "UI",
            "date": "2016-02-28T16:41:41+0000",
            "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
    }
}
```

   b. Using the POST method, start the process of deploying an application using the selected provider. For example, if we use the website: https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:sort=launchTime we can check the created instances.
   c. For *OneComponentApp,* we copy the public URL of the created virtual machine and copy it to a new window in our browser. Then the AWS web page should be displayed. The AWS installation is included in the *AmazonEC2.xmi Provider* Model. We do not have to install it manually.
   d. Check that *TwoComponentApp* is created and installed on one provider (i.e. AWS). One virtual machine is created for the database and a second virtual machine is created for an application. We can open the AWS web page (as before), copy the IP of the application VM, and enhance it as follows: http://Virtual_machine_IP:9999/demo/all. If we open this URL on another web browser tab, the application website should be displayed.
   e. For the Test Cases where the user selects two different providers, one virtual machine is created on one provider and a second virtual machine on another provider. We can use as a first provider for example AWS, and as a second provider Omistack: https://omistack.e-technik.uni-ulm.de/dashboard/auth/login/). Then we can use the link: http://Virtual_machine_IP:9999/demo/all (as before) to deploy.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

5.2 How to execute Test Cases with attached CP Models

This section presents the Quality Assurance set of steps to be followed when executing Test Cases with attached CP models.

1. Upload the CP model's .xmi file to the machine's file system where the CP Generator resides (for example: melodic@<VM's IP>/logs/UserTest_temp)
2. Execute the procedure by sending POST message to the CP Generator to the constraintProblemSolutionFromFile URL.
   a. By using the tools SoapUI or Postman execute the following step:
      - Create the REST project with URL
        http://<VM's IP>:8093/constraintProblemSolutionFromFile and body:

Sample body

```
{     "applicationId": "Test",
"fileModelsPath": "/logs/UserTest_temp/CpModelName.xmi",
             "watermark": {
             "user": "ebankowska",
             "system": "test",
             "date": "2017-09-28T16:41:41+0000",
             "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
    }
}
```

   b. Proceed to melodic@<VM's IP>/logs and display the log by using for example vi, i.e., *vi cpsolver.log.*

6 Test Cases

The test process starts at the very beginning of a project life cycle. During the Analysis and Design phase, the test team starts producing a Test Plan, as this should be prepared as early as possible. A Test Plan contains test cases, which are described in detail in the "D5.10 Quality Assurance Guide" deliverable [3]. The Test Plan should also explicate the dependencies between the Test Cases (if needed), by clarifying which Test Cases should be executed before others. This whole exercise has three main benefits:

- It allows the test team to understand the system to be developed
- It serves as a review of the system specifications and requirements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- It eases solving issues, as all parties (the test team and the development teams) have the same base data (the test data; input parameters for test cases, necessary to execute test cases and to reproduce bugs, if occurring during test case execution).

A test case describes how to perform a specific test. The test case includes a set of test data, pre-conditions, expected results and post-conditions targeting a certain implementation, developed for a specific purpose or for the condition mapping to the test such as the execution of a program path, or to verify compliance with a specific requirement. Test cases are created by the test team, either its members or the test leader.

6.1 Executed Test Cases

This section presents two lists of executed Test Cases for the Melodic releases 1.0 and 1.5, correspondingly. In particular, Table 7 shows the identifier of the executed test cases for release 1.0, including a summary of each test case (where 'T' equals True/A Positive Test Case, and 'F' equals False/A Negative Test Case) in terms of its task corresponding identifier and name[4], as well as its execution status. The content of the group column is explained in Section 6.2. Similarly, Table 8 shows the executed test cases for release 1.5. For the first release, 20 Test Cases were executed with status "passed" and 1 with status "failed" (due to an old version of Cloudiator). For release 1.5, 21 Test Cases were executed with status "passed" and one with status "failed".

Table 7: List of all executed Test Cases for the first release

KEY	Summary	Type	Group	Status
MT-4	T1.1[T] Installation and deployment of a simple application on one Cloud Provider	Functional	Initial deployment	Passed
MT-8	T1.1[F] Installation and deployment of a simple application on one Cloud Provider	Functional	Initial deployment	Passed
MT-5	T1.2[T] Installation and deployment of a two-component application on two different cloud providers	Functional	Initial deployment	Passed

[4] The 'Key' and 'Summary' fields correspond to the internal identification and naming used in the Melodic team's JIRA.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

MT-9	T1.2[F] Installation and deployment of a two-component application on two different cloud providers	Functional	Initial deployment	Passed
MT-6	T1.3[T] Installation and deployment of a two-component application on two different Cloud Providers	Functional	Initial deployment	Passed
MT-51	T5.1[T] Linear constraints and optimization solving – CP Solver	Functional	Reasoning related	Passed
MT-54	T5.4[T] Non-linear constraints and optimization solving – CP Solver	Functional	Reasoning related	Passed
MT-31	T6.1[T] Temporary unavailability of particular components	Functional	Reasoning related	Passed
MT-42	T6.1[F] Temporary unavailability of particular components	Functional	Reasoning related	Passed
MT-44	T6.3[F] Temporary unavailability of Cloud Provider	Functional	Reasoning related	Passed
MT-34	T6.4[T] High Availability Component configuration	Functional	Reasoning related	Passed
MT-41	T6.4[F] High Availability Component configuration	Functional	Reasoning related	Passed
MT-45	T6.5[T] Response time while solving allocation problems	Non-functional	Performance	Passed
MT-60	T7.5[T] User authentication	Other non-functional	Logging	Passed
MT-66	T7.6[T] Logging within MELODIC platform	Other non-functional	Unified administration procedures	Passed
MT-57	T8.2[T] Removing user	Other non-functional	User management	Passed
MT-58	T8.3[T] Updating user password	Other non-functional	User management	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

MT-59	T8.4[T] Updating user profile	Other non-functional	User management	Passed
MT-61	T8.5[T] Unified starting, stopping and restarting of MELODIC platform	Other non-functional	Unified administration procedures	Passed
MT-63	T8.7[T] Executing backup	Other non-functional	Unified administration procedures	Passed
MT-56	T8.1[T] Adding user	Other non-functional	User management	Failed

Table 8: List of all executed Test Cases for the Release 1.5

KEY	Summary	Type	Group	Status
MT-87	T2.3[T] Composite metric detection (1st Level Event Processing)	Functional	Metric management	Passed
MT-88	T2.3[F] Composite metric detection (1st Level Event Processing)	Functional	Metric management	Passed
MT-89	T2.5[T] Composite metric detection (2nd Level Event Processing)	Functional	Metric management	Passed
MT-90	T2.5[F] Composite metric detection (2nd Level Event Processing)	Functional	Metric management	Passed
MT-91	T2.6[T] Custom raw metrics collection – FCR model	Functional	Metric management	Passed
MT-93	T4.3[T] Global reconfiguration rules testing – FCR	Functional	Global reconfiguration	Passed
MT-73	T5.6[T] Utility function – FCR	Functional	Reasoning related	Passed
MT-32	T6.2[T] Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery	Non-functional	Fault handling	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

MT-43	T6.2[F] Temporary unavailability of BPM	Non-functional	Fault handling	Passed
MT-48	T6.8[T] Counting Resource Overhead of Melodic instance introduced over its host	Non-functional	Performance	Failed
MT-76	T9.1[T] [Model Editor] Login	Functional	User Interface	Passed
MT-77	T9.2[T] [Model Editor] Logout	Functional	User Interface	Passed
MT-78	T9.3[T] [Model Editor] Schema Management – Create Concept	Functional	User Interface	Passed
MT-79	T9.4[T] [Model Editor] Schema Management – Create Property	Functional	User Interface	Passed
MT-80	T9.5[T] [Model Editor] Schema Management – Delete Property	Functional	User Interface	Passed
MT-81	T9.6[T] [Model Editor] Schema Management – Delete Concept	Functional	User Interface	Passed
MT-82	T9.7[T] [Model Editor] Schema Management – Update Model Repository	Functional	User Interface	Passed
MT-83	T9.8[T] [Model Editor] Schema Management – Update Local Repository	Functional	User Interface	Passed
MT-84	T9.9[T] [Model Editor] Schema Management – Clear Local Repository	Functional	User Interface	Passed
MT-85	T9.10[T] [Model Editor] Schema Management – Import from XMI	Functional	User Interface	Passed
MT-86	T9.11[T] [Model Editor] Schema Management – Export to XMI	Functional	User Interface	Passed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6.2 All Test Cases created during Release 1.0 and 1.5

In total, during release 1.0, 60 Test Cases were created, and during release 1.5 the number was 22. Every Test Case was categorised into a particular group, as shown in Table 9. The groups are further described below the table.

Table 9: Categories of test cases in first release and release 1.5

Functional Test Cases groups	Test Cases created during release 1.0 (summary)	Test Cases created during release 1.5 (summary)
Initial Deployment	17	
Global/Local Reconfiguration	6	1
Metric Management	6	6
Reasoning Related	5	4
User Interface		11
Total	34	22
Non-functional Test Cases groups		
Fault Handling	8	
Performance	3	
Security	4	
Other (logging, user management, unified administration procedures)	11	
Total	26	

- **Initial Deployment** – This group contains all scenarios related to the initial deployment of an application in the Melodic platform.
- **Local Reconfiguration** – Local reconfiguration means that the reconfiguration of the application or its parts occurs in a certain cloud, and is based on the scalability rules defined in SRL in the CAMEL model of the application. For this type of test cases, the selected modules of Upperware and Executionware are tested (the SRL adapter and Cloudiator).
- **Metric Management** – Metric management means the collection, processing (aggregation), storage and delivery of raw and composite metrics, as well as CAMEL events based on these metrics. For this test cases group, the Executionware modules are the mostly tested elements (e.g., the Metric Collector), but due to the installation of an

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

application – which has also a definition of corresponding metrics and events – the key modules of the Upperware and Executionware are tested too.

- Reasoning Related – Reasoning maps to the capability to find an optimal deployment solution for the application at hand based on the requirements that have been posed for it in the application's CAMEL model (requirement sub-model). The Test Cases are focused on isolated tests over each particular solver. Mostly, the Upperware modules are tested here (i.e., the CP Generator, the Meta Solver, the CP Solver, the MILP Solver, and the LA Solver).
- Fault Handling – This section presents scenarios related to the Fault Handling Test Case group. Fault handling maps to the reliability of the system and the ability to properly handle technical failures, crashes and external system inaccessibility. The group's test scenarios present the most common situations and focus on the verification of the (application) deployment process, the global and local reconfiguration, as well as the deployment reasoning, thus involving and focusing on all components of the Melodic system.
- Security Related – This section presents Test Cases related to testing the security in the Melodic system and in its communication with external systems. Security, for the purpose of these tests, means authentication as well as authorization of methods invocation between components, especially methods exposed on ESB, and the usage of secure, cryptographically protected communications protocols. The security scope, the security mechanism requirements and design, as well as the security testing cases related to advanced security scenarios, will be covered in more detail in the "D5.03 Security requirements and design" deliverable [2].
- Performance – In general, performance testing of Cloud-applications is done similarly to web applications. In performance testing of a web application, it is up to the tester to decide the performance related parameters (mainly latency and throughput), based on specific user provided requirements. However, it must be kept in mind that application performance is dependent on the user's perception (e.g., for a latency sensitive web-application, a lower response time is desirable). As the types of such web applications are different, the user requirements could also be differentiated.
- Global Reconfiguration – Global reconfiguration is the reconfiguration of the application at a global scope, where a new solution is applied globally for the whole application and not only on its specific parts (in contrast to local reconfiguration). Such a reconfiguration is applied mainly by the Upperware module and especially the solvers in the presence of one or more contextual changes (for example, new metric measurements, provider offering modifications, etc.).
- Other non-functional testing related Test Cases – This group covers Test Cases related to backup and recovery along with platform user addition/removal, monitoring, logging, administration and maintenance tasks – aspects which are relevant to guarantee the reliability and recoverability of the system as well as the overall administration. For the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

releases 1.0 and 1.5, only the Cloudiator UI is available, so the testing of user management in these releases are limited to the Cloudiator UI.

The whole content of the executed Test Cases during release 1.0 and 1.5 can be found Appendix A, 'Test Cases Release 1.0 and Release 1.5'.

7 Bugs

During the development and testing of a system, bugs can be found. A bug is a defect in a component or system that can cause that component or system not to perform its required function. It constitutes a deviation of the component or system from its expected delivery, service or result. Bugs are reported to support their correction, which can then enable the system to subsequently work as planned and expected. Bugs are corrected by developers and checked by testers.

7.1 Reported bugs

During release 1.0, 23 bugs were detected and reported, whereof most of them where immediately corrected; No blocking issues (issues with the "highest" priority, which can block progress of testing) were detected during the tests, while six important bugs with priority "high" have been fixed and closed. Four unresolved errors with priority "high" were not immediately fixed due to issues related to the CP generator and CDO configuration. These were all resolved for release 1.5.
During release 1.5, 28 bugs were reported, where of the most important bugs, with priority "high" or "highest", have been fixed. There are still four unresolved bugs from this release with "low" and "medium" priority. All together, the quality of the Melodic platform implementation for release 1.0 and release 1.5 was evaluated positively. Table 10 and Table 11 list the reported bugs for release 1.0 and release 1.5, respectively.

Table 10: Reported bugs during release 1.0

Key	Priority	Summary	Status
MEL-331	Medium	Provided ports are not opened on cloud provider's machines	Closed
MEL-328	Medium	TwoComponentApp doesn't deploy using QA1 environment	Closed
MEL-323	Medium	Problem with reconfiguration graph	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

MEL-315	Medium	No allowed boot sources for UULM Openstack	Closed
MEL-294	Medium	Log4j2 could not find a logging implementation - Colosseum logs	Closed
MEL-293	Medium	TwoComponentApp: creating 3 instances instead of 2 (deploying process)	Closed
MEL-285	High	Problem during creating application on AWS	Closed
MEL-284	Medium	Application Component Instance ApacheAppInstance_1_0 does not seem to be working	Closed
MEL-267	High	CPSolver - OneComponentApp - Problem is infeasible	Closed
MEL-252	Medium	Issue with connecting provider requirement with particular requirement set	Closed
MEL-248	Medium	Generated CP model not visible in log files	Closed
MEL-247	Medium	Process blocks in random places	Closed
MEL-246	Low	Incorrectly displayed time in the CP generator log	Closed
MEL-245	Low	Incorrectly displayed time in the Mule log	Closed
MEL-230	High	Missing version in pom.xml for io.github.cloudiator:common	Closed
MEL- 82	High	Failed Meta-solver tests	Closed
MEL-297	High	UULM openstack instances running very slow	Closed
MEL-295	Medium	org.eclipse.emf.cdo.util.CommitException: Rollback in HibernateStore: org.hibernate.PropertyValueException	Closed
MEL-330	High	CDO client error - "Not active: PackageRegistry"	Closed. Fixed in release 1.5
MEL-329	High	Improper working CDO	Closed. Fixed in release 1.5
MEL-327	Low	New User cannot be added to the Cloudiator UI	To Do
MEL-249	Medium	Not active BranchManager during generating CpModel	Closed. Fixed in release 1.5
MEL-251	Medium	NullPointer during generating log	Closed. Fixed in release 1.5

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Table 11: Reported bugs during release 1.5

Key	Priority	Summary	Status
MEL-429	Low	Problem with filtering nodeCandidates by location name	To do
MEL-438	Medium	Different type in swagger and in response from findNodeCandidates	To do
MEL-437	Medium	Deploying new machine with 'TwoComponentApp' or 'FCR' results in 'Error during generating CpModel'	Closed
MEL-436	Medium	Unable to create a VM with specified number of disk storage	To do
MEL-404	Low	CDO server oxygen build failes	To do
MEL-407	High	Adapter error during deploying CAMEL .xmi with changed DB requirements	Closed
MEL-411	Medium	The application doesn't run properly (Web page doesn't display)	Closed
MEL-408	High	Cloudiator doesn't delete VM template	Closed
MEL-420	High	Mule doesn't work	Closed
MEL-403	High	CP Generator: HttpServerErrorException: 500 Response code 400 mapped as failure	Closed
MEL-409	Low	The application doesn't run properly (after reconfiguration)	Closed
MEL-415	Medium	CP Generator - Error during generating CP Model for CAMEL Model without metric (e.g. TwoComponentApp)	Closed
MEL-412	Highest	Error saving CP solution	Closed
MEL-345	Highest	Problem with Colosseum-client	Closed
MEL-414	Medium	FCR CAMEL- Process restarts at the end of deploying	Closed
MEL-416	Medium	Problem with reconfiguration	Closed
MEL-428	Medium	Location eu-west-1c in cloud does not exist in Colosseum	Closed
MEL-405	High	Adapter is not up and running (Q1 Machine) [Protocol version 19 does not match expected version 34]	Closed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

MEL-406	Medium	Problems with QA1 machine	Closed
MEL-434	Medium	VMS server could not retrieve VM info from Cloudiator	Closed
MEL-431	Medium	Metric messages lost in translation	Closed
MEL-432	Medium	Metrics are not collected via Esper 1st level	Closed
MEL-433	Medium	Visor process terminates	Closed
MEL-413	High	Different type of machine than the chosen one	Closed
MEL-402	Medium	Application doesn't see function: melosic.setSession	Closed
MEL-435	Medium	Problem with storing data to cache under key	Closed
MEL-439	Medium	cpSolver is missing 'utilityGenerator.properties' file	Closed
MEL-430	Medium	Unable to fully deploy app on t2.micro machine types	In progress

8 Summary

This deliverable presents the Melodic 1.0 and 1.5 integration releases and the corresponding test environments. The following information has been described in the document:

1. An introduction to the integration relates
2. A high-level overview of the Melodic architecture
3. The testing environments for Melodic release 1.0 and 1.5
4. A Melodic platform installation guide
5. Testing guides on how to execute Test Cases with attached CAMEL Models and CP Models
6. Test Cases executed for Melodic release 1.0 and 1.5
7. An overview of bugs found during development and testing of Melodic release 1.0 and 1.5

Release 1.5 is an additional intermediate release aimed at reducing the span between the initial integration release of Melodic (release 1.0) and the Melodic platform prototype (release 2.0) by verifying the functionality of the system and the its components responsible for the correct operation of the application and its functionality.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

9 References

[1] F. Zahid et al., "D2.2 Architecture and initial feature definitions", The Melodic H2020 Project Deliverable D2.2, 2018.

[2] P. Skrzypek et al., "D5.03 Security requirements & design", The Melodic H2020 Project Deliverable D5.03, 2018.

[3] M. Jakubczyk at al., "D5.10 Quality Assurance Guide", The Melodic H2020 Project Deliverable D5.10, 2017

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Appendix A - Test Cases Release 1.0 and Release 1.5

A.1 Release 1.0

T1.1[T] Installation and deployment of a simple application on one Cloud Provider.

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to the given Cloud Provider.
5. **Complete CAMEL model of simple application** (which includes the definition of the application components and their installation/maintenance scripts). Simple application – one component application, installed as a unix process (no container), in one virtual machine *(please, refer to the attached .xmi file).*
6. CAMEL model of given Cloud Provider prepared and registered in the MELODIC platform with at least one virtual machine offer provided.
7. There should be a proper configuration of the virtual machine both in CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed MELODIC by using following steps:
   - Use command: ssh melodic@<VM IP>
2. Download and run:
   cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar
   from:
   https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload models:
   - cpGenerator-functionTypes.xmi,
   - cpGenerator-locations.xmi,
   - cpGenerator-providerTypes.xmi,
   - cpGenerator-operatingSystems.xmi

   Into:
   "/home/user/models" directory
4. Upload Provider *AmazonEC2.xmi* into "/home/models/upperware-models/fms" directory
5. Upload CAMEL model *OneComponentApp.xmi* into "/home/models" directory
6. Using **SoapUI tool** execute following steps:
   - Create new **REST** project with URL:
     http://<VM IP>:8088/api/frontend/deploymentProcess

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

- Using POST method start process

7. Start deploying of application

For each step, the status of the executed action should be positive.

Expected Results:

1. Virtual machine on the selected Cloud Provider should be created.
2. The sole component of the simple application should be installed on that machine.
3. Correctly installed and working Apache server.
4. **The application should be run properly** (*Apache web page is properly displayed).*

T1.1[F] Installation and deployment of a simple application on one Cloud Provider

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to the given Cloud Provider.
5. **Complete CAMEL model of simple application** (which includes the definition of the application components and their installation/maintenance scripts). Simple application – one component application, installed as a ***"False" test cases "False" test cases*** *should include improperly created CAMEL model (it means CAMEL model .xmi with errors).* Please, refer to the attached .xmi file.
6. CAMEL model of given Cloud Provider prepared and registered in the MELODIC platform with at least one virtual machine offer provided. There should be a proper configuration of the virtual machine both in CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps to Complete:

1. Login to the machine with installed MELODIC by using following steps:
   - Use command: ssh melodic@<VM IP>
   - Download and run *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar* from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
   - Upload CAMEL model *OneComponentFalseApp.xmi* into "/home/models" directory

For upload CAMEL model step, the status of the executed action should be negative (uploading model *OneComponentFalse.xmi* should not be stored into CDO).

Expected results:

1. Virtual machine on the selected Cloud Provider should not be created.
2. The sole component of the simple application *should not be installed* on that machine.
3. Apache server does not install and working correctly.
4. The application should not be run properly *(Apache web page is not displayed).*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T1.2[T] Installation and deployment of a two-component application on one Cloud Provider

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to the given Cloud Provider.
5. **Complete CAMEL model of the two-component application** (*which includes the definition of these two components and their installation/maintenance scripts).* One component can map to the main business logic of the application and the other to the underlying database used. An application is Wordpress which also includes an underlying MySQL database *(please, refer to the attached .xmi file).*
6. CAMEL model of given Cloud Provider prepared with at least one virtual machine offer provided. There should be a proper configuration of the virtual machine both in CAMEL Provider model (see step 2) and on the Cloud Provider side. The configurations should be aligned.

Steps to Complete:

1. Login to the machine with installed MELODIC by using following steps:
    a. Use command: ssh melodic@<VM IP>
2. Download and run *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar* from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi, cpGenerator-providerTypes.xmi, cpGenerator-operatingSystems.xmi into "/home/user/models" directory
4. Upload Provider model *AmazonEC2.xmi* into "/home/models/upperware-models/fms" directory
5. Upload CAMEL model *TwoComponentApp.xmi* into "/home/models" directory
6. Using **SoapUI tool** execute following steps:
    a. Create new **REST** project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
    b. Using POST method start process (body attached in attachments)
7. Start deploying of application:
    a. Open page: http://"public ID of created Virtual Machine":9999/demo/all

For each step, the status of the executed action should be positive.

Expected results:

1. *Two virtual machine instances* (of the same VM flavour/offering) should be created using the selected Cloud Provider.
2. The application should be installed on those VM instances (one business logic component instance should be installed *on the first VM instance and the database component instance should be installed on the second VM instance).*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3. **The application should run properly** which actually involves that proper communication between application components has been established (*The Wordpress page from database should be displayed correctly).*

T1.2[F] Installation and deployment of a two-component application on one Cloud Provider

Input Conditions:

1. Installed and configured MELODIC platform
2. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to the given Cloud Provider.
5. **Complete CAMEL model of the two-component application** (*which includes the definition of these two components and their installation/maintenance scripts).* One component can map to the main business logic of the application and the other to the underlying database used. An application is Wordpress which also includes an underlying MySQL database. ***"False" test cases** should include improperly created CAMEL model (it means CAMEL model .xmi with errors).* Please, refer to the attached .xmi file.
6. CAMEL model of given Cloud Provider prepared with at least one virtual machine offer provided. There should be a proper configuration of the virtual machine both in CAMEL Provider model (see step 2) and on the Cloud Provider side. The configurations should be aligned.

Steps to Complete:

1. Login to the machine with installed MELODIC by using following steps:
    a. Use command: ssh melodic@<VM IP>
2. Download and run *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar* from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload CAMEL model *TwoComponentFalseApp.xmi* into "/home/models" directory
4. For upload CAMEL model step, the status of the executed action should be negative (uploading model *TwoComponentFalse.xmi* should not be stored into CDO).

Expected results:

1. *Two virtual machine instances* (of the same VM flavour/offering) should not be created using the selected Cloud Provider.
2. The application should not be installed on those VM instances (one business logic component instance should not be installed *on the first VM instance and the database component instance should not be installed on the second VM instance).*
3. The application should not run properly (The Wordpress page from database should not be displayed).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T1.3[T] Installation and deployment of a two-component application on two different Cloud Providers

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. At least two cloud providers integrated with MELODIC platform, where the user has provided his/her own credentials for both of them (included in CAMEL model- cloud credentials).
3. Meta solver configured to use CP Solver for that case.
4. Cloudiator properly connected to given Cloud Providers.
5. **Complete CAMEL model of the two-component application** *(which includes the definition of these two components and their installation/maintenance scripts)*. One component can map to the main business logic of the application and the other to the underlying database used. An application is Wordpress which also includes an underlying MySQL database *(please, refer to the attached .xmi file).*
6. CAMEL model of each Cloud Provider prepared with at least one virtual machine offer provided per each provider.
7. There should be a proper configuration of the virtual machine both in CAMEL Providers model and on the Cloud Providers sides.
8. There should be a requirement in the application CAMEL model to use different Cloud Providers (this could be done in different ways; for example, by placing a location requirement that is then referenced in the virtual machine requirement set.

Steps to Reproduce:

1. Login to the machine with installed MELODIC by using following steps:
    a. Use command: ssh melodic@<VM IP>
    b. Use command: **. .profile**
2. Download and run *cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar* from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
3. Upload models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi, cpGenerator-providerTypes.xmi, cpGenerator-operatingSystems.xmi into "/home/user/models" directory
4. Upload Providers *AmazonEC2.xmi* and *OpenStackUlm.xmi* into "/home/models/upperware-models/fms" directory
5. Upload CAMEL model *TwoComponentApp.xmi* into "/home/models" directory
6. Using **SoapUI tool** execute following steps:
    a. Create new **REST** project with URL: http://<VM IP>/api/frontend/deploymentProcess
    b. Using POST method start process (body attached in attachments)
7. Start deploying of application:
    a. Open page: http://"public ID of created Virtual Machine":9999/demo/all
8. *gs-mysql-data-0.1.0-non-executable.jar* can be download from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/

For each step, the status of the executed action should be positive.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. Two VM instances should be created, one instance per each Cloud Provider.
2. *The application should be installed on those 2 VM instances* (one business logic component instance should be installed on the first VM instance and the database component instance should be installed on the second VM instance).
3. **The application should be run properly** (The Wordpress web page from database should be displayed correctly).

T5.1[T] Linear constraints and optimization solving – CP Solver

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.
2. Provide the CP model of given optimization problem – as described in the table with reference data (please see attachments). Model should be uploaded to CDO.

Steps to Complete:

1. To run the test directly on CP Solver following steps need to be performed:
2. Prepare XMI file with CP model (please, refer to the attachments)
3. Put the above XMI to machine's file system where CP Generator resides (for example /tmp)
4. Execute CP Generator solving procedure by sending POST message to CP Generator to constraintProblemSolutionFromFile URI.

Sample body

```
{"applicationId": "Dam",
fileModelsPath": "/tmp/cpGenerator_cpm_cp.xmi",
"watermark": {
        "user": "ebankowska",
        "system": "test",
        "date": "2017-09-28T16:41:41+0000",
        "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
          }
}
```

Expected results:

1. Optimal solution is found.
2. CP model is updated in CDO.
3. In CP Solver log, the main constraint problem and its optimal solution found should be logged.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T5.4[T] Non-linear constraints and optimization solving – CP Solver

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.
2. Provide CP model of given non-linear optimization problem – as described in the table with reference data (please see attachments).

Steps to Complete:

1. To run the test directly on CP Solver following steps need to be performed:
2. Prepare XMI file with CP model (please, refer to the attachments)
3. Put the above XMI to machine's file system where CP Generator resides (for example /tmp)
4. Execute CP Generator solving procedure by sending POST message to CP Generator to constraintProblemSolutionFromFile URI.

Sample body

```
{   "applicationId": "Dam",
fileModelsPath": "/tmp/cpGenerator_cpm_cp.xmi",
"watermark": {
"user": "ebankowska",
        "system": "test",
        "date": "2017-09-28T16:41:41+0000",
        "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
        }
}
```

Expected results:

1. Optimal solution is found
2. CP model is updated in CDO.
3. In CP Solver log, the main constraint problem and its optimal solution found should be logged.

T6.1[T] Temporary unavailability of particular components

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. At least one cloud provider has been integrated with MELODIC platform, and user has supplied respective credentials for this provider.
3. Meta solver configured to use CP Solver for that case.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4. Cloudiator properly connected to given Cloud Provider.
5. **Complete CAMEL model of a simple application** (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in one VM. An application is the Apache Webserver installed using a standard installation command *(please, refer to the attached .xmi)*.
6. CAMEL model of given Cloud Provider prepared with at least one virtual machine offer included.
7. There should be a proper configuration of the virtual machine both in CAMEL Provider model (see step 2) and on the Cloud Provider side. The configurations should be aligned.

Steps to Complete:

1. Login to the machine with installed MELODIC by using following steps:
    a. Use command: ssh melodic@<IP VM>
2. Upload models into "models" directory
3. Upload Cloud Provider into "models/upperware-models/fms" directory
4. Upload CAMEL model *OneComponentApp.xmi* into "models" directory
5. On MELODIC platform:
    a. Using command **dps** check CONTAINER ID for 1 generator
    b. Using command **sudo docker stop CONTAINER ID1** to stop first generator
6. After stopping first or second generator as soon as possible start process by using SoapUI (Process must be deployed before generator is up and running)
7. Use cd /logs and display generator.log

For each step, the status of the executed action should be positive.

Expected Results:

The results are listed more or less in order of occurrence with the exception that the action to restart a component takes place in between a pair of stated actions and requires the re-execution of the first step in the action pair:

1. Proper error message with information about stopped component inaccessibility should be logged.
2. VM (instance) on the selected Cloud Provider should be created.
3. The simple application should be installed on that VM (instance).
4. **The application should be run properly** (the web server's web page should be displayed properly).

T6.4[T] High Availability Component configuration

Input Conditions:

1. *Installed and configured MELODIC platform*, without any application related artefacts.
2. HA configuration of components; each of the following components should be installed with two instances, with HA configuration on ESB:
    a. CP Generator
    b. Meta solver

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

   c. CP Solver
   d. Solver to deployment
   e. Adapter/Plan Generator
3. At least one cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model- cloud credentials).
4. Meta solver has been configured to use the CP solver for that case.
5. Cloudiator properly connected to given Cloud Provider
6. **Complete CAMEL model of a simple application** (which includes the definition of one component and its installation/maintenance scripts). Simple, one component application installed as a unix process (no container) in one virtual machine. An application is an Apache Webserver installed using a standard installation command *(please, refer to the attached .xmi).*
7. CAMEL model of given Cloud Provider prepared with at least one virtual machine offering included.
8. There should be a proper configuration of the virtual machine both in CAMEL Provider model and on the Cloud Provider side.

Steps To Complete:

1. Login to the machine with installed MELODIC by using following steps:
   a. Use command: ssh melodic@<IP VM>
2. Upload models into "models" directory
3. Upload Cloud Provider into "models/upperware-models/fms" directory
4. Upload CAMEL model *OneComponentApp.xmi* into "models" directory
5. On MELODIC platform:
   a. Edit *melodic.yml* and change generator replicas to 2
   b. Use command **ddeploy**
   c. Using command **dps** check CONTAINER ID for 2 generators
   d. Using command **sudo docker stop CONTAINER ID1** to stop first generator
   e. Using command **sudo docker stop CONTAINER ID2** to stop second generator
6. After stopping first or second generator as soon as possible start process by using SoapUI (Process must be deployed before generator is up and running)
7. Use cd /logs and display generator.log
8. Check logs for two generator.1. and generator.2.

For each step, the status of the executed action should be positive.

Expected results:

1. Proper error message with information about fall-back due to component inaccessibility should be logged.
2. A VM (instance) on the selected Cloud Provider should be created.
3. The simple application should be installed on that VM (instance).
4. **The application should be run properly** (web server's web page should be displayed properly).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T6.5[T] Response time while solving allocation problems

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefact.
2. Each of the below components should be installed with single instances: CP Generator, CDO Server, Meta solver, CP Solver, Solver to deployment and Adapter.
3. At least one Cloud provider integrated with the MELODIC platform; the user credentials for this provider should have also been supplied (included in CAMEL model-Cloud credentials).
4. Meta solver configured to use CP solver for that case.
5. Cloudiator properly connected to a given Cloud Provider.

Complete CAMEL model of a simple application (which includes the definition of the application components and their installation/maintenance scripts). A simple application is a one component application, installed as a Unix process (no container), in one virtual machine.
CAMEL model of given Cloud provider prepared with the number of VM offerings included. CAMEL model of given Cloud Provider prepared and registered in the MELODIC platform with at least one virtual machine offer provided. There should be a proper configuration of the virtual machine both in CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed MELODIC by using following steps:
    a. Use command: ssh melodic@<IP VM>
    b. Download and run cdo-uploader-1.0.0-SNAPSHOT-jar-with-dependencies.jar from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data
    c. Upload models: cpGenerator-functionTypes.xmi, cpGenerator-locations.xmi cpGeneratorproviderTypes.xmi, cpGenerator-operatingSystems.xmi into "/home/user/models" directory
2. Upload Provider AmazonEC2.xmi into "/home/models/upperware-models/fms" directory
3. Upload CAMEL model OneComponentApp.xmi into "/home/models" directory
4. Using SoapUI tool execute following steps:
    a. Create new REST project with URL: http://5.249.145.169:8088/api/frontend/deploymentProcess
    b. Using POST method start process
    c. Start deploying of application
    d. Check solver logs

For each step, the status of the executed action should be positive.

Expected Results:

1. Virtual machine on the selected Cloud Provider should be created.
2. The sole component of the simple application should be installed on that machine. Correctly installed and working application.
3. The application should be run properly.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4. Error messages due to component inaccessibility or any other issues should be logged.
5. Execution time of each Upperware components must be logged.
6. Response time of each Upperware components must be logged.

T7.10[T] Logging within MELODIC platform

Input Conditions:

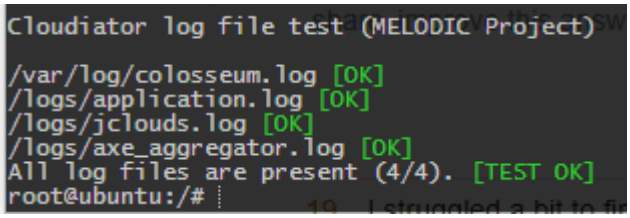
1. Installed and configured MELODIC platform, without any application related artefacts.

Steps To Complete:

1. Login to MELODIC/Cloudiator VM with root credentials.
2. Execute the attached script './test_logging.sh'

Expected results:

1. Output of test script equals the following screenshot



T7.2[T] Removing user

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.

Steps To Complete:

1. Log into the frontend User section of executionware_ui of Cloudiator with existing administrator credentials (i.e., http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/frontendUser
2. Press 'delete' button located in the user's row.

Expected results:

1. User profile should not be listed anymore on:
    a. http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/frontendUser
2. Login with user is not possible any longer.

T7.3[T] Updating user password

Input Conditions:

1. Running MELODIC/Cloudiator installation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps to Complete:

1. Login to Cloudiator VM as root.
2. Download 'password-generator-1.0-SNAPSHOT-jar-with-dependencies.jar' i.e.,
   wget http://../password-generator-1.0-SNAPSHOT-jar-with-dependencies.jar
3. Execute command 'java -jar password-generator-1.0-SNAPSHOT-jar-with-dependencies.jar'
4. Note generated password, salt and hashed password
5. Execute 'mysql -uroot -p'$password' -e "UPDATE colosseum.FrontendUser SETpassword = '$password', salt = '$salt' WHERE firstName = '$name' and lastName = '$lastName';"' i.e., for default user John Doe -> execute 'mysql -uroot -p'$password' -e "UPDATE colosseum.FrontendUser SET password = '$password', salt = '$salt' WHERE firstName = 'John' and lastName = 'Doe';"'
6. Adjust Tenant_FrontendUser if necessary, INSERT INTO Tenant_FrontendUser (tenants_id,frontendUsers_id) VALUES (1, 2).

Expected results:

1. Login with changed credentials is possible:
   a. i.e., with user 'John Doe' to:
      http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui

T7.4[T] Updating user profile

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.

Steps to Complete:

1. Log into the frontend User section of executionware_ui of Cloudiator with existing administrator credentials (i.e., http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/frontendUser)
2. Press 'edit' button located in the user's row
3. Adapt the user information accordingly
   a. firstName
   b. lastName
   c. mail
4. Submit the completed form using the 'Submit' button

Expected results:

1. User profile information on http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/frontendUser matched previous changes

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T7.6[T] Unified starting, stopping and restarting of MELODIC platform

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.

Steps to Complete:

1. Login to MELODIC/Cloudiator VM with root credentials
2. Start the Cloudiatorwith the following commands:
    a. rm /opt/cloudiator/colosseum-0.2.0-SNAPSHOT/RUNNING_PID
    b. sudo sh /opt/cloudiator/start_colosseum.sh&
3. Stop the Cloudiator with the following commands run 'sudo pkill -f 'java.*colosseum''
4. Restart the Cloudiator
   rm /opt/cloudiator/colosseum-0.2.0-SNAPSHOT/RUNNING_PID
   sudo sh /opt/cloudiator/start_colosseum.sh&

Expected results:

1. If overall status is STARTED: MELODIC/Cloudiator can be used as usual
2. If overall status is STOPPED: MELODIC/Cloudiator

T7.5[T] User authentication

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.

Steps to Complete:

1. Log into the frontend User section of executionware_ui of Cloudiator with existing administrator credentials (i.e., http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/frontendUser)
2. Create a new user
3. Browse to http://cloudiator.melodic.dcsresearch.cas.de/executionware_ui/#/tenant)
    a. Press 'add' button
    b. Define a tenant name
    c. Add the previously added user as a tenant member
4. Submit the completed form using the 'Submit' button

Expected results:

1. Newly created user is now able to login with the tenant of the administrator that created this user.

T7.8[T] Executing backup

Input Conditions:

1. Installed and configured MELODIC platform, without any application related artefacts.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps to Complete:

1. Login to Cloudiator VM as root
2. Execute command 'cd /opt&& tar -zcvf ~/cloudiator-backup.tar.gz cloudiator'
3. Download backup file 'cloudiator-backup.tar.gz'
4. Execute command 'sudo mysqldump -uroot -pmelodic colosseum > ~/cloudiator-backup.sql'
5. Download backup file 'cloudiator-backup.sql'
6. Execute command 'cd ~&& zcvf tar -/var/lib/docker/volumes/melodicStack_Conf/_data'
7. Download backup file 'melodic-backup.tar.gz'

Expected results:

1. Backup files 'cloudiator-backup.tar.gz', 'cloudiator-backup.sql' and 'melodic-backup.tar.gz' can be used to restore the Cloudiator installation on a newly installed Cloudiator.

A.2 Release 1.5

T2.3[T] Composite metric detection (1st Level Event Processing)

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider is integrated with Melodic platform for which the user has supplied the respective credentials (included in Camel model- cloud credentials).
3. Cloudiator is properly connected to given Cloud Provider.
4. (optional) CAMEL model of FCR application with definition of custom metrics (please, refer to the attached .xmi).
5. At least jdk 1.7 should be installed on the VM.

Steps To Complete:

1. Login to the machine with installed Melodic
2. On the VM execute the deploy of FCR application
3. Send test raw events though REST using e.g. postman to the IP: VM_IP:8083:
   (send at least 2 events in 1 minute) under the event topic ResponseTime with the following payload:

```
{" metricValue": 90,
    "vmName": "FCR.VM1",
    "cloudName": "Openstack1",
    "componentName" : " FCR",
    "level": 0,
     "timestamp": 1224235434
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. After 1 minute, go to the logs folder of mule-standalone-3.9.0, execute cat person.log and check for one complex event produced by the first level Esper (the certain test rule produces the average response time every 1 minute). This complex event will have:
    a. as a value the average of all the values of raw events intercepted during this time window (note: this will be a sliding window),
    b. the level set to 1 and
    c. the timestamp updated

T2.3[F] Composite metric detection (1st Level Event Processing)

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider is integrated with Melodic platform for which the user has supplied the respective credentials (included in Camel model- cloud credentials).
3. Cloudiator is properly connected to given Cloud Provider.
4. (optional) CAMEL model simple of application (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in one VM. An application is the Apache Webserver installed using a standard installation command) with definition of raw system/built-in metrics (please, refer to the attached .xmi).
5. At least jdk 1.7 should be installed on the VM.

Steps To Complete:

1. Login to the machine with the deployed component
2. On the VM execute the deploy1_app.sh that configures, installs and starts Mule and Esper. This deployment script will set up a mechanism that is able to intercept raw events (manually sent through REST) and calculate the average value (e.g. average response time) every one minute based on the incoming raw events).
3. Don't send any test raw events for one minute

Expected results:

1. After 1 minute, go to logs folder of mule-standalone-3.9.0, execute cat person.log. No raw or complex events should be available.

T2.5[T] Composite metric detection (2nd Level Event Processing)

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider is integrated with the Melodic platform for which the user has supplied the respective credentials (included in Camel model- cloud credentials).
3. Cloudiator is properly connected to given Cloud Provider.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

4. (optional) CAMEL model of simple application (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in two VMs (i.e. two component instances). An application is the Apache Webserver installed using a standard installation command) with definition of raw system/built-in metrics (please, refer to the attached .xmi).
5. At least jdk 1.7 should be installed on each of the two VMs.

Steps To Complete:

1. Login to the two machines with the deployed component instances
2. On the first VM execute the deploy2_app.sh that configures, installs and starts an AMQ Broker that will receive not only local events, but also events coming from other VMs in the same cloud and an Esper instance able to detect first & second level complex event patters (i.e. average response time detected from all VMs on a certain cloud (i.e. the two VMs in this testing scenario)).
3. On the second VM perform the following:
    a. update the IP mentioned in the deploy1_app.sh using the IP used for the first VM (i.e. se the proper IP in this command: sudo ./mule -M-Dhost.broker="tcp:// VM1_IP:61616")
    b. execute the deploy1_app.sh that configures, installs and starts Mule and first level Esper and connects to the first VM.
4. Check that the 2nd VM is properly connected to the second level broker
    a. Login to the first VM and execute netstat -an|grep 61616
5. Send test raw events though REST using postman to the first VM using the following IP: VM1_IP:8083 in postman. Send (at least 2 events in 1 minute) under the event topic *ResponseTime* with the following payload:

```
{" metricValue_": 90,_
  _"vmName": "OneComponentApp.VM1",_
  _"cloudName": "Openstack1",_
    "componentName" : " OneComponent",
    "level": 0,
    "timestamp": 1224235434
 }
```

6. Send test raw events though REST using postman to the 2nd VM using the following IP: VM2_IP:8083 in postman. Send (at least 2 events in 1 minute) under the event topic *ResponseTime* with the following payload:

```
{" metricValue_": 80,_
  _"vmName": "OneComponentApp.VM2",_
  _"cloudName": "Openstack1",_
    "componentName" : " OneComponent",
    "level": 0,
    "timestamp": 1224235434
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. In step 4, during the checking of the established connectivity with the 2nd level AMQ message broker you should get in the registered clients the IP of the second VM
2. After steps 5,6 and once 1 minute has passed, go to the logs folder of mule-standalone-3.9.0 (Execute cat person.log) of the 1st VM (where the 2nd level broker is hosted) and check for one complex event produced by the 2nd level Esper (the certain test rule produces the average response time every 1 minute from all VMs on a certain cloud (the two VMs in our testing scenario). This complex event should have:
    a. Level value equals to 2
    b. vmName value equals to OneComponentApp.VM1, OneComponentApp.VM2
    c. as a value the average of all the values of raw events intercepted during this time window (note: this will be a sliding window),
    d. the timestamp updated

T2.5[F] Composite metric detection (2nd Level Event Processing)

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider is integrated with the Melodic platform for which the user has supplied the respective credentials (included in Camel model- cloud credentials).
3. Cloudiator is properly connected to given Cloud Provider.
4. (optional) CAMEL model of simple application (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in two VMs (i.e. two component instances). An application is the Apache Webserver installed using a standard installation command) with definition of raw system/built-in metrics (please, refer to the attached .xmi).
5. At least jdk 1.7 should be installed on each of the two VMs.

Steps To Complete:

1. Login to the two machines with the deployed component instances
2. On the first VM execute the deploy2_app.sh that configures, installs and starts an AMQ Broker that will receive not only local events, but also events coming from other VMs in the same cloud and an Esper instance able to detect first & second level complex event patters (i.e. average response time detected from all VMs on a certain cloud (i.e. the two VMs in this testing scenario)).
3. On the second VM perform the following:
    a. update the IP mentioned in the deploy1_app.sh using the IP used for the first VM (i.e. se the proper IP in this command: sudo ./mule -M-Dhost.broker="tcp:// VM1_IP:61616")
    b. execute the deploy1_app.sh that configures, installs and starts Mule and first level Esper and connects to the first VM.
4. Check that the 2nd VM is properly connected to the second level broker
    a. Login to the first VM and execute netstat -an|grep 61616
5. Don't send any test raw events for one minute

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. In step 4, during the checking of the established connectivity with the 2nd level AMQ message broker you should get in the registered clients the IP of the second VM
2. After steps 5 and once 1 minute has passed, go to the logs folder of mule-standalone-3.9.0 (Execute cat person.log) of the 1st VM (where the 2nd level broker is hosted). No event should be registered in the logs.

T2.6[T] Custom raw metrics collection - FCR model

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider is integrated with Melodic platform for which the user has supplied the respective credentials (included in Camel model- cloud credentials).
3. Cloudiator is properly connected to given Cloud Providers.
4. CAMEL model of FCR application with definition of the custom raw metrics.
   Model .xmi can be downloaded from:
   https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/FCR

Steps To Complete:

1. Login to the machine with installed Melodic by using following steps:
   Use command: ssh melodic@<VM IP>
2. Upload Camel model *FCR.xmi* into "models" directory
3. Using SoapUI tool execute following steps:
   a. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
   b. Using POST method start process
4. Start deploying of application

For each step, the status of the executed action should be positive.

Expected results:

1. Values of the given type raw metric(s) should be stored in TS database(s) on those VMs/nodes on which the respective component of the application to be measured resides.
2. These measurements/values are possibly stored in the CDO and some subscribers (Meta Solver) might be informed about them.

T4.3[T] Global reconfiguration rules testing – FCR

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider integrated with Melodic platform for which the respective user credentials have been supplied.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

3. Cloudiator properly connected to given Cloud Providers.
4. There exists at least one cloud provider with at least two offering satisfying the requirements posed by the user.
5. CAMEL model of each Cloud Provider prepared and uploaded to the platform with at least two virtual machine offers provided with different cost parameter.
6. Complete CAMEL model of a FCR application. The CAMEL model should include definition of events and metrics needed to execute the particular test case.
   Model .xmi can be downloaded from:
   https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/FCR

Steps To Complete:

1. Login to the machine with installed Melodic by using following steps:
   Use command: ssh melodic@<VM IP>
2. Upload Camel model *FCR.xmi* into "models" directory
3. Using SoapUI tool execute following steps:
   a. Create new REST project with URL: http://<VM IP>:8088/api/frontend/deploymentProcess
   b. Using POST method start process
4. Deploy application
5. Check FCR application IP (ip-app)
6. Launch Active MQ: http://ip-app:8161
   http://ip-app:8161/ -> topics -> notice Response Time
   (response time - measured values by Visor sending to the 1st level Esper)
7. Launch Active MQ http://<virtual_machine_IP>:8161
   login -> topics -> notice Average Response Time and AverageResponseTimeHigh1m
   (values sending from 1 level Esper to 2nd level Esper)
8. To activate metrics - disable java security for ip_app:8087
9. Run: http://ip-app:8087/SiC/index.jsp
10. Observe sending metrics for both sides of ActiveMQ - first for application, next between Esper levels.
    There should also appears metric: AverageResponseTimeHigh1m then open metasolver log and check if reconfigurations has been initiated.

For each step, the status of the executed action should be positive.

Expected results:

1. Application should be reconfigured according to defined SLOs.
2. Application should work properly; this means that its web page should be properly displayed (continuing the previous example with an Apache web server).

T5.6[T] Utility function – FCR

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. Provide CP model of given utility function for FCR.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Login to the machine with installed MELODIC
2. Deploy FCR.xmi
3. After deploying FCR.xmi, proceed to the /logs/cp_model_upperware-models/ and check full name of generated CP model .xmi (for example: FCR1519638005224.xmi). The name of the full CP model will be displayed in generator.log. For example:
   INFO 1 — [SimpleAsyncTaskExecutor-1] e.p.u.p.g.o.GenerationOrchestrator : ** CP Model Id: upperware-models/FCR1519638005224
4. Restart MELODIC platform by using *drestart* command
5. Deploy FCR<number>.xmi by
   Using tools: SoapUI or Postman execute following steps:
   Create REST project with: URL: <VM's IP>:8093/constraintProblemSolutionFromFile
   Sample body:

```
{"applicationId": "FCR",
 "fileModelsPath": "/logs/cp_model_upperware-models/FCR<number>.xmi",
 "nodeCandidatesFilePath": "/logs/node_candidates_upperware-models/FCR<number>",
 "useExternalOptimizer": "true",
 "watermark": {
        "user": "test",
        "system": "UI",
        "date": "2017-11-23T16:41:41+0000",
        "uuid": "fb6280ec-1ab8-11e7-93ae-92361f002671"
  }
 }
```

6. Proceed to the: melodic@<VM's IP>/logs and display cp solver log

Expected results:

1. Optimal solution is found:
   cardinality = 1 (the cheapest offer)
2. CP model is updated in CDO.
3. In Utility Generator log, the constraint problem and its derived solution are logged.

T6.2[T] Temporary unavailability of BPM - verifying proper system behaviour after BPM recovery

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider has been integrated with Melodic platform, while the user has supplied his/her credentials for this provider.
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly connected to given Cloud Provider.
5. BPM component is stopped.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

6. Complete CAMEL model of a simple application (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in one VM. An application is the Apache Webserver installed using a standard installation command. CAMEL model of OneComponentApp can be found in:
   https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/OneComponentApp

Steps To Complete:

1. Login to the machine with installed Melodic by using following steps:
   Use command: ssh melodic@<VM IP>
2. Upload Camel model *OneComponentApp.xmi* into "models" directory
3. Using SoapUI tool execute following steps:
   a. Create new REST project with URL: http://<VM's IP>/api/frontend/deploymentProcess
   b. Using POST method start process
4. After deploying of application:
   a. Use command *dps* to display all IDs of components
   b. Use command *sudo docker stop CONTAINER ID1* to stop BPM
5. Display BPM log

For each step, the status of the executed action should be positive.

Expected results:

1. Proper error message with information about jBPM inaccessibility should be logged.
2. A VM (instance) on the selected Cloud Provider should be created.
3. The simple application should be installed on that VM (instance).
4. The application should be run properly (the web server's website is displayed properly).

T6.2[F] Temporary unavailability of jBPM

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider has been integrated with Melodic platform, while the user has supplied his/her credentials for this provider.
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly connected to given Cloud Provider.
5. jBPM component is stopped.
6. Complete CAMEL model of a simple application (which includes the definition of one component and its installation/maintenance scripts). The simple, one-component application is installed as a unix process (no container) in one VM. An application is the Apache Webserver installed using a standard installation command.
   "False" test cases should include improperly created CAMEL model (it means a CAMEL model .xmi with errors). Please, refer to the attached .xmi file.
7. CAMEL model of given Cloud Provider prepared with at least one virtual machine offer included.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

8. There should be a proper configuration of the virtual machine both in CAMEL Provider model (see step 2) and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the machine with installed Melodic by using following steps:
   Use command: ssh melodic@<VM IP>
2. Upload models into "models" directory
3. Upload Cloud Provider into "models/upperware-models/fms" directory
4. Upload Camel model *JBPMFalseApp.xmi* into "models" directory
   For upload Camel model step, the status of the executed action should be negative (uploading model *JBPMFalse.xmi* should not be stored into CDO).

Expected results:

1. Improper error message with information about jBPM inaccessibility should be logged.
2. A VM (instance) on the selected Cloud Provider should not be created.
3. The simple application should not be installed on that VM (instance).
4. The application should not run properly (the web server's website is not displayed properly).

T6.8[T] Counting Resource Overhead of Melodic instance introduced over its host

Input Conditions:

1. Installed and configured Melodic platform, without any application related artefacts.
2. At least one cloud provider integrated with the Melodic platform; the user credentials for this provider should have also been supplied (included in Camel model- cloud credentials).
3. Meta solver configured to use CP solver for that case.
4. Cloudiator properly connected to the given Cloud Provider.
5. Complete CAMEL model of simple application (which includes the definition of the application components and their installation/maintenance scripts). Simple application – one component application, installed as a unix process (no container), in one virtual machine.
   CAMEL model of OneComponentApp can be found in:
   https://bitbucket.7bulls.eu/projects/TST/repos/melodic/browse/TestCases/OneComponentApp
6. CAMEL model of given Cloud Provider prepared and registered in the Melodic platform with at least one virtual machine offer provided.
7. There should be a proper configuration of the virtual machine both in CAMEL Provider model and on the Cloud Provider side. The configurations should be aligned.

Steps To Complete:

1. Login to the Melodic machine: ssh melodic@<VM's IP>
2. Deploy first OneComponentApp VM on AWS by using Melodic platform:
   a. Download and run cdo-uploader-1.0.1-SNAPSHOT-jar-with-dependencies.jar from: https://s3.console.aws.amazon.com/s3/buckets/melodic.testing.data/
   b. Upload Camel model OneComponentApp.xmi into "/home/models" directory.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

c. Using SoapUI tool execute following steps:
    i. Create new REST project with URL:
        http://<VM's IP>:8088/api/frontend/deploymentProcess
        http://5.249.145.169:8088/api/frontend/deploymentProcess
    ii. Using POST method start process
d. Start deploying of application.
e. Connect to the Melodic VM and use these commands to display information (Two files: *basicResources.txt* and *htop.html* will be created.):
    i. wget https://s3-eu-west-1.amazonaws.com/melodic.testing.data/checkResources.sh
    ii. chmod +x checkResources.sh
    iii. ./checkResources.sh

3. Next, start a new Virtual Machine manually to install the application under test (without using the Melodic platform):
    a. On AWS tap Melodic VM then select: *Actions/Launch More Like This*
    b. Connect to the second VM (without Melodic) and use commands:
        i. to download: sudo apt-get update
        ii. to configure: sudo apt-get --assume-yes install apache2
        iii. To start: sudo service apache2 start
    c. Connect to the Manual VM and also start:
        i. wget https://s3-eu-west-1.amazonaws.com/melodic.testing.data/checkResources.sh
        ii. chmod +x checkResources.sh
        iii. ./checkResources.sh
4. We compare overhead (resource consumptions) for both cases.

For each step, the status of the executed action should be positive.

Expected results:

1. Virtual machine on the selected Cloud Provider should be created.
2. The sole component of the simple application should be installed on that machine.
3. Correctly installed and working Apache server.
4. The application should be run properly (Apache web page is properly displayed).
5. The log file must include the CPU usage, Memory page swap/RAM usage, process usage and network/disk I/O for comparison.
6. Error messages due to component inaccessibility or any other issues should be logged.
7. For further comparison, the same application must be executed (with success) in an independent VM, and it must also produce another usage/trace report for comparison with Melodic environment.

T9.1[T] [Model Editor] Login

Input Conditions:

1. Metadata Schema Editor (also referred to as 'MUSE') up and running and login page loaded

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Steps To Complete:

1. Fill in user name
2. Fill in user password
3. Hit 'Login' button

Expected results:

1. Welcome page is shown

T9.2[T] [Model Editor] Logout

Input Conditions:

1. User is logged in into editor

Steps To Complete:

1. user opens 'hamburger' menu
2. user hits 'Logout' button

Expected results:

1. user is logged out and gets redirected to login page

T9.3[T] [Model Editor] Schema Management - Create Concept

Input Conditions:

1. user is on page 'Melodic Metadata Schema Management'

Steps To Complete:

1. user focuses root element of model tree (left side)
2. user initiates creation of Concept
3. user defines unique (non-existing) name, e.g., 'test'
4. user hits "Save Changes"

Expected results:

1. newly created concept 'test' is available in the model tree

T9.4[T] [Model Editor] Schema Management - Create Property

Input Conditions:

1. user is on page 'Melodic Metadata Schema Management'

Steps To Complete:

1. user focuses root element of model tree (left side)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

2. user initiates creation of Property
3. user defines unique (non-existing) name, e.g., 'testProperty'
4. user hits "Save Changes"

Expected results:

1. newly created Property 'testProperty' is available in the model tree

T9.5[T] [Model Editor] Schema Management - Delete Property

Input Conditions:

1. user is on page 'Melodic Metadata Schema Management'

Steps To Complete:

1. user selects Property to be deleted, e.g., 'testProperty'
2. user hits 'Delete' button

Expected results:

1. Property is not present anymore in the model tree

T9.6[T] [Model Editor] Schema Management - Delete Concept

Input Conditions:

1. user is on page 'Melodic Metadata Schema Management'

Steps To Complete:

1. user selects Concept to be deleted, e.g., 'test'
2. user hits 'Delete' button

Expected results:

1. Concept is not present anymore in the model tree

T9.7[T] [Model Editor] Schema Management - Update Model Repository

Input Conditions:

1. user created a random property with unique name
2. user is on welcome page of MUSE

Steps To Complete:

1. user hits button ''Local repos. -> Model repos.'
2. user hits "Model repos. -> Local repos.' button

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

Expected results:

1. random property is present (in model tree; still present and was not deleted)

T9.8[T] [Model Editor] Schema Management - Update Local Repository

Input Conditions:

1. user is logged in
2. meanwhile, a new CDO entry (e.g., property with unique name 'testProperty') is added

Steps To Complete:

1. user refreshes model tree

Expected results:

1. user finds newly added property 'testProperty'

T9.9[T] [Model Editor] Schema Management - Clear Local Repository

Input Conditions:

1. user is logged in
2. local model is present (e.g., model was loaded) with at least one entry

Steps To Complete:

1. user hits "Clear Local repos.' button

Expected results:

1. Metadata Schema Management indicates empty local model repository

T9.10[T] [Model Editor] Schema Management - Import from XMI

Input Conditions:

1. user is logged in
2. local repository is empty (nothing loaded)

Steps To Complete:

1. user hits 'Import from XMI' button
2. user double clicks on XMI area and uploads file (e.g., 'metadata-schema-0.0.2.xmi') when prompted to do so
3. user hits button 'Model repos. -> Local repos.'

Expected results:

1. local repository (e.g., model tree) indicates that items are present

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664

T9.11[T] [Model Editor] Schema Management - Export to XMI

Input Conditions:

1. user is logged in
2. local repository is not empty

Steps To Complete:

1. user hits button 'Export Metadata in XMI'

Expected results:

1. download of file starts

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731664