Climate Change Initiative Extension (CCI+) Phase 1

New Essential Climate Variables (NEW ECVS)

High Resolution Land Cover ECV (HR_LandCover_cci)

System Verification Report

(SVR)

Prepared by:

Università degli Studi di Trento
Fondazione Bruno Kessler
Université Catholique de Louvain
Università degli Studi di Pavia
Università degli Studi di Genova
Politecnico di Milano
Université de Versailles Saint Quentin
CREAF
e-GEOS s.p.a.
Planetek Italia
GeoVille

Changelog

Issue	Changes	Date
0.1	First version.	01/03/2020
1.0	Final version including all the test on the system	17/04/2020

Detailed Change Record

Issue	RID	Description of discrepancy	Sections	Change
N/A	N/A	N/A	N/A	N/A

Contents

1 Introduction

1.1 Executive summary

Within the European Space Agency (ESA), the Climate Change Initiative (CCI) is a global monitoring program which aims to provide long-term satellite-based products to serve the climate modelling and climate user community. Permafrost has been selected as one of the Essential Climate Variables (ECVs) which are elaborated during Phase 1 of CCI+ (2018-2021).

Following the activities of user requirements updating according to Climate User Community and other users' consultations, the Consortium has defined the related HRLC products requirements accounting for technical constraints such as main data sources available, spatial and temporal coverage, software and tools for quality control. This High Resolution Land Cover (HRLC) System Specification Document defines the system architecture and the description of the first version of the system that will generate the CCI+ HRLC products over the areas of interests.

This document outlines the system verification procedures and results for the system with respect to its capabilities to meet the challenges described in paragraph 2.1.

A test scenario is defined for the system verification. Two key aspects of the processing system are tested: 1. Capability to scale automatically 2. The capability to easily design pipelines. The verification procedure makes use of known processing steps as the final processors (steps) for the generation of HRLC products are still in the integration phase. The goal of the verification is to ensure that the system is capable to on-board generic processing steps.

The scaling capability is then guaranteed also by the fact that the system is deployed on a Cloud and thus can access to a nearly infinite resource pool.

1.2 Purpose and scope

The HRLC System Verification Report defines the criteria to meet the challenges defined in the SSD and verifies the availability of them in the system.

Input to this document are the Tender Specification [AD2] and the other Applicable Document and, in particular, to meet the challenges defined in System Specification Document (SSD)

The document is organized with the following contents:

- The description of the challenges of the system (2.1) and of the test environment (2.2)
- The description of a verification scenario (3.1)
- The design of the test for the verification (3.2)
- The description and execution of the test cases (3.3)

1.3 Applicable documents

Ref.	Title, Issue/Rev, Date, ID
[AD1]	CCI HR Technical Proposal, v1.1, 16/03/2018
[AD2]	CCI Extension (CCI+) Phase 1 – New ECVs – Statement of Work, v1.3, 22/08/2017, ESA-CCI-PRGM-EOPS-SW-17-0032
[AD3]	Data Standards Requirements for CCI Data Producers, v2.1, 02/08/2019, CCI-PRGM-EOPS-TN-13-0009
[AD4]	User Requirements Document, v1.1, 12/04/2019, CCI_HRLC_Ph1-D1.1_URD
[AD5]	Product Specification Document, v1.0, CCI_HRLC_Ph1-PSD
[AD6]	Data Access Requirement Documentm v1.0, CCI_HRLC_Ph1-DARD
[AD7]	System Requirement Document v2.0, CCI_HRLC_Ph1-SRD

[AD8] System Specification Document v1.0, CCI_HRLC-Ph1-SSD

1.4 Reference documents

Ref. **Title, Issue/Rev, Date, ID**

[RD1] The Global Climate Observing System: Implementation Needs, 01/10/2016, GCOS-200

1.5 Acronyms and abbreviations

API Application Programming Interface
AOI Area Of Interest
ARD Analysis Ready Data
AWS Amazon Web Services
CCI Climate Change Initiative
CRC Climate Research Community
CMUG Climate Modelling User Group
DIAS Data and Information Access Services
ECV Essential Climate Variables
ESM Earth System Models
EVI Enhanced Vegetation Index
FTP File Transfer Protocol
GCOS Global Climate Observing System
GDPR General Data Protection Regulation
GIS Geographical Information System
HR High Resolution
IaaS Infrastructure as a Service
L1C Level-1C
L2A Level-2A
LAI Leaf Area Index
LaSRC Landsat Surface Reflectance Code
LC Land Cover
LCC Land Cover Change
LCCS Land Cover Coverage Classification System
LCML Land Cover Meta Language
LCZ Local Climate Zone
LEDAPS Landsat Ecosystem Disturbance Adaptive Processing System
LSCE Laboratoire des Sciences du Climat et de l'Environnement
MR Medium Resolution
NDVI Normalized Difference Vegetation Index
OGC Open Geospatial Consortium
OWS OGC Web Services
PFT Plant Functional Type

RS	Remote Sensing
SAR	Synthetic Aperture Radar
SFT	Surface Functional Type
SRD	Software Requirements Document
SSD	Software Specification Document
SVR	Software Verification Report
TOA	Top Of Atmosphere
URD	User Requirements Document
VM	Virtual meeting
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WP	Work Package

2 Challenges for System Verification

2.1 Main challenges from SSD

The platform prototype scope is to bring the results of the research activities to a pre-operational level by scaling up the processing capacity in order to allow the production of massive land cover mosaics following GCOS requirements. In practice, having to deal with several TB of data (hundreds) means that the concept of pre-operational is not applicable and that the system must provide a huge capability to scale the processing. For this reason, some constrains and requirements coming from the SRD are dealing with the capability of the IaaS platform chosen for the execution of the production.

2.1.1 Challenge 1: Orchestration of resources and IaaS

The orchestration platform, described in the following paragraphs, is based on the concept of big-data processing and reactive pipeline execution (Reactive Manifesto, https://www.reactivemanifesto.org/en) and is based on Max-ICS technology (Max-ICS by Earthlab Louxembourg, http://www-max-ics.earthlab.lu/) which is part of e-GEOS CLEOS platform (currently used internally but soon to be released as a service).

The challenge is to perform a test of scalable processing under known conditions this means that a known processor will be used to do some basic data processing to verify the scalability. The verification is done by

- Verifying the activation of multiple resources as requested by the designed test on a single node execution

2.1.2 Challenge 2: Flexibility in the pipeline development

Another important point is the capability of the platform to manage easily changes in the pipeline with minimal manual intervention. In particular, the prototype platform will put the basis for future enhancement by allowing easy link between the research activity and the production activity.

The overall process starts from the research activity, using Conda environment (https://conda.io/). This activity provides as output the code (and its updates) on the internal GitLab. Then, each processor generates a processor as described in SRD that is managed by the orchestrator to run a pipeline.

The challenge is to design and implement a pipeline under known conditions linking processing nodes. The verification is done by:

- Verifying the capability to design a pipeline of nodes
- Verifying the capability to run the pipeline and obtain the final results

2.1.3 Challenge 3: Delivery of results

In addition, a general OGC server is added to the processing platform that is used for the access to the large amount of data. The server allows to search for data collections using OGC CSW interface and to visualize/download data (e.g. HRLC products) using OGC WMS/WCS services.

The challenge is to implement a software component able to present different type of geospatial data and in particular of the type raster and vector, the verification is done by:

- Verifying the capability to show Metadata type of data (Vector encoding) as OGC Service and through a Web Interface
- Verifying the capability to show raster data of different type, specifically Sentinel 2 data (True Color Image) and a final product classification

2.2 Verification environment

The environment is fully configured on a Cloud and is designed to exploit the capability to be elastic of a cloud. In the following table we give some technical details of the software and configuration info where applicable.

The configuration is made in region eu-central-1 (frankfurt) as Sentinel data are only hosted on this region. The test environment is equivalent to the final environment for what concerns the Orchestrator scope as it can scale using all resources of the cloud. For the delivery system it is downgraded (to avoid wasting of resources) even if is made of on-demand resources that can be scaled anytime. The reference for the instances is given at https://aws.amazon.com/ec2/instance-types/?nc1=h_ls.

Component	Software	Configuration
Orchestrator	Max-ICS, http://www-max-ics.earthlab.lu/	Orchestrator: 3 t3.2xlarge (8 vCPU 32.0 GB) instances on AWS Worker: r4.2xlarge
IaaS	AWS S3, https://aws.amazon.com/s3/?nc1=h_ls	Bucket: S3://ccihrlc
	AWS EC2, https://aws.amazon.com/ec2/?nc1=h_ls	Region: eu-central-1
Code Repository	GITLAB, https://about.gitlab.com/	Not Applicable
OGC Server	Gesoerver, http://geoserver.org/	Geoserver and Postgres Database: t3a.2xlarge 8 vCPU 32.0 GB

Table 1: Environment configuration

3 System Verification

The objective of this paragraph is to describe a known scenario in terms of processors (with well-known and tested input and output) and to use the scenario to design test cases for the system according to the verification scenario.

3.1 Design of the verification scenarios

In version 1 of this SVR we present the verification test as applicable to the system described in the SSD. Detailed verification tests and results including the processor integration will be included in the following versions of the document.

The scenario for the verification of the system is described in the following points:

- Creation of the pipeline and execution of a processing of Sentinel 1 SLC data to generate Amplitude products.
- The process is discovered using OpenEO Discovery API
- The execution is triggered using OpenEO Execution API.
- Products to be delivered demonstrating the capability to deliver different products both raster and vector:
    - Sentinel 2 catalogue
    - Sentinel 2 L2A True Color Image
    - Raster Classification

3.2 Design of the verification tests

Scenarios for the verification are taken from the SSD and in particular from the Enterprise Viewpoint as the interest is to demonstrate the capability of the system to meet user needs and the technical challenges. The following user are then introduced:

1. **User of final products role** - it is the user that access the delivery services to visualise the products and access to them with available interfaces, the use cases are
2. **User of processing service role** - it is the user that access the available interfaces (API/CLI) to execute a processing service, the use case is
3. **Expert user role** - it is the user that access to the available interfaces (API/UI) to deploy a new processing service and create a pipeline using those available

The following table links the challenges with the users and define the verification test. The numbering of the tests is explained as follow:

- The first number TEST-1-1 identifies the challenge number as defined in paragraph 2.1 so TEST-1-1 is related to the Challenge 1
- The second number TEST-1-1 identifies the incremental number of test on the challenge so TEST-1-1 is the first test of the challenge

Code	User	Test Name	Verification Objective
TEST-1-1	User of processing service	OpenEO Process Discovery API	Verify the availability of OpenEO API for Process Discovery
TEST-1-2	User of processing service	OpenEO Process Execution API	Verify the availability of OpenEO API for Process Execution
TEST-1-3	User of processing service	Activation of multiple resources	Verify the activation of multiple resources as requested by the designed test on a single node execution
TEST-1-4	User of processing service	Pipeline execution monitoring	Verify the monitoring of execution of a pipeline from the interface
TEST-2-1	Expert user role	Pipeline Design	Verifying the capability to design a pipeline of nodes
TEST-2-2	Expert user role	Pipeline Run	Verifying the capability to run the pipeline and obtain the final results
TEST-3-1	User of final product	Metadata Display	Verifying the capability to show Metadata type of data (Vector encoding) as OGC Service and through a Web Interface
TEST-3-2	User of final product	Products Display	Verifying the capability to show raster data of different type, specifically Sentinel 2 data (True Color Image) and a final product classification

Table 2: Organisation of test with respect to users and challenges

3.3 Verification tests

3.3.1 TEST-1.1: OpenEO Process Discovery API

TEST-1.1: OpenEO Process Discovery API
Initial conditions: 1. The OpenEO API is up and running 2. A Process is registered in the system
Test execution procedure: 1. The operator send the OpenEO request according to the specification defined for discovery of jobs using GET with no: **Error! Hyperlink reference not valid.** where the url is masked internally 2. The server responds with a JSON document

Test result:
The requests is executed correctly as shown in the following JSON Snippet
{
"processes": [{
"id": "apply",
"summary": "Perform SAR geocoding on images",
"description": "",
"categories": [
"Sentinel 1"
],
"parameters": [{
"name": "data",
"description": "image",
"schema": {
"type": "object",
"subtype": "image"
}
},
…

3.3.2 TEST-1.2: OpenEO Process Creation API

TEST-1.1: OpenEO Process Creation API
Initial conditions: 1. The OpenEO API is up and running 2. A Process is registered in the system

Test execution procedure:

1. The operator send the OpenEO request according to the specification defined for creation of batch job: https://localhost/api/0.4/jobs **Error! Hyperlink reference not valid.** where the url is masked internally


```
{
    "title": "NDVI based on Sentinel 2",
    "description": "Perform SAR geocoding on images ",
    "process_graph": {
        "dc": {
            "process_id": "load_collection",
            "arguments": {
                "id": "Sentinel-1",
                "spatial_extent": {
                    "west": 16.1,
                    "east": 16.6,
                    "north": 48.6,
                    "south": 47.2
                },
                "temporal_extent": [
                    "2018-01-01",
                    "2018-02-01"
                ]
            }
        },
        "sargeocoding": {
            "process_id": "sargeocoding",
            "description": "",
            "arguments": {
                "data": {
                    "from_node": "dc"
                },
                "polarization": [
                    "All"
                ]
            }
        },
        "result": true
```

```
        }
}
```

2. The server responds with a 200 status code sending back a 201 status code with the following information:
    - Location URL of the created resource: ex. https://openeo.org/api/v0.4/jobs/123
    - OpenEO-Identifier: ex. 123

Test result:

The request is executed correctly. The process then can be executed sending a POST request to the created resource: https://openeo.org/api/v0.4/jobs/123/results

3.3.3 TEST-1.3: Activation of multiple resources

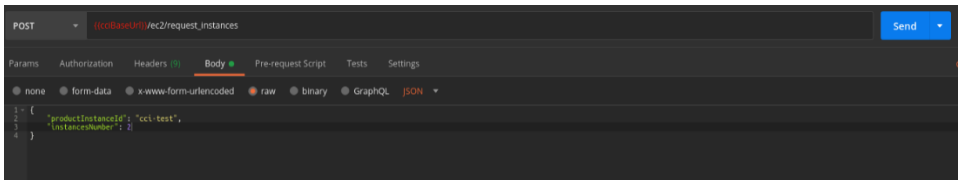
TEST-1.3: Activation of multiple resources

Initial conditions:

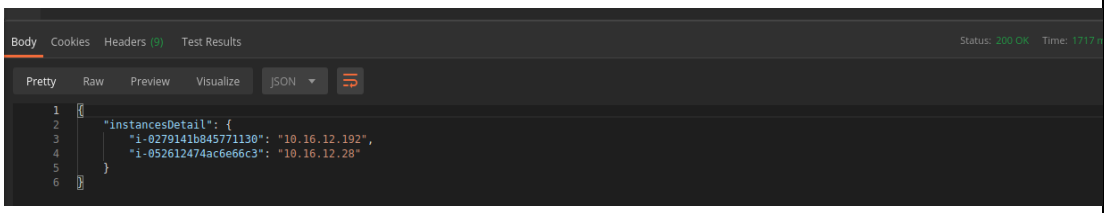
1. The OpenEO API is up and running
2. AWS API is up and running

Test execution procedure:

1. The user sends a POST request to the API to activate multiple resources

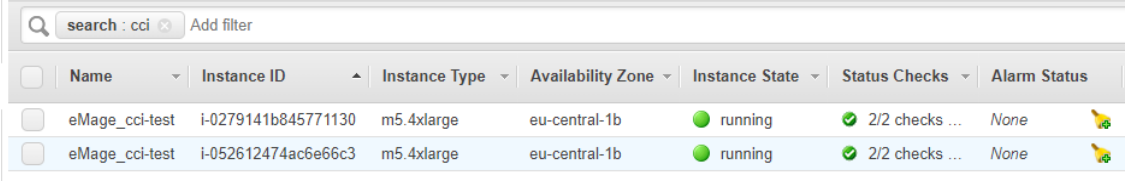


2. The user checks that the response status is “200 OK” and the response body contains the IDs of activated resources



Test result:

The request is executed correctly as shown in the following screenshot in AWS Console:

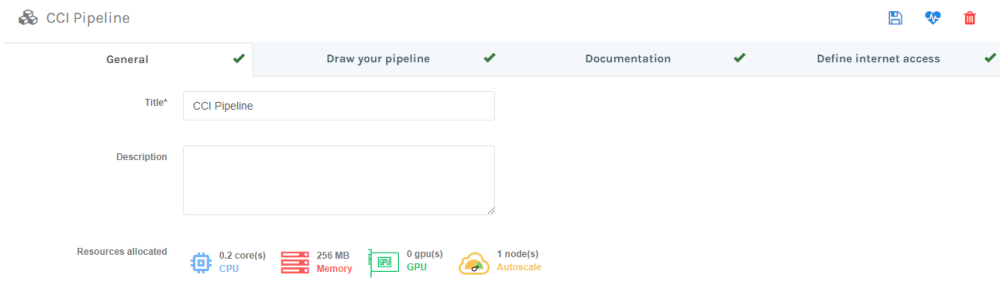


3.3.4 TEST-1.4.1: Pipeline execution monitoring using Service API

TEST-1.4.1: Pipeline execution monitoring using Service API
Initial conditions: 1. The OpenEO API is up and running 2. The pipeline is executing at least one job
Test execution procedure: 1. The user sends a GET request to the API endpoint /jobs with the running job ID 2. The user checks the “status” field in the JSON response

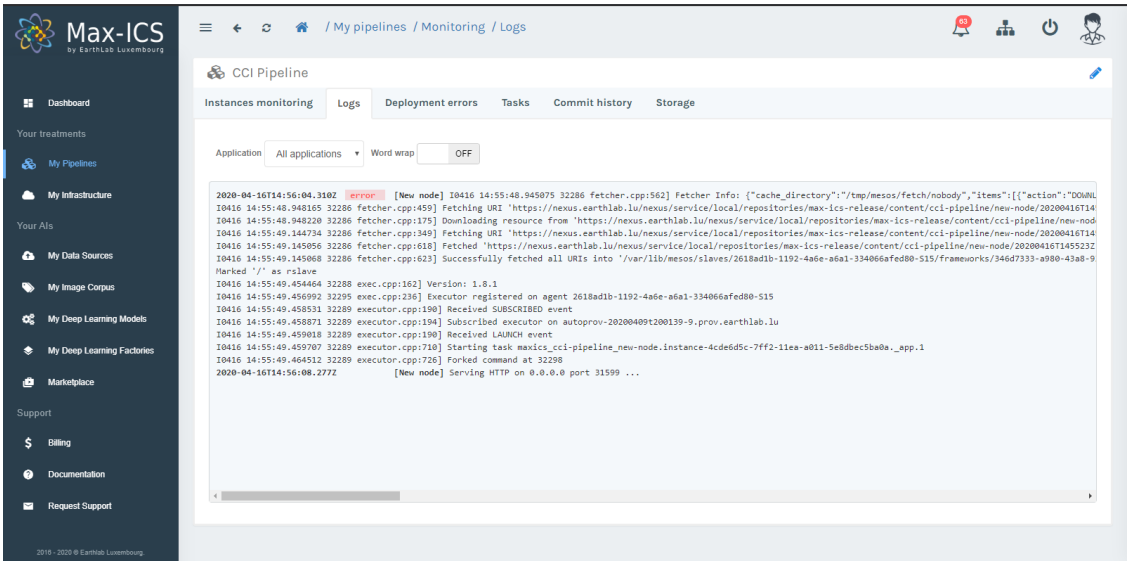
Test result:
The requests is executed correctly as shown in the following JSON
{
"product_instance_id": "test-product-instance-id",
"node_id": "test-node-id",
"service_id": "test-service-id",
"items": [
"test-item-1",
"test-item-2"
],
"status": "started",
"delivered_links": []
}

3.3.5 TEST-1.4.2: Pipeline execution monitoring using Max-ICS logs

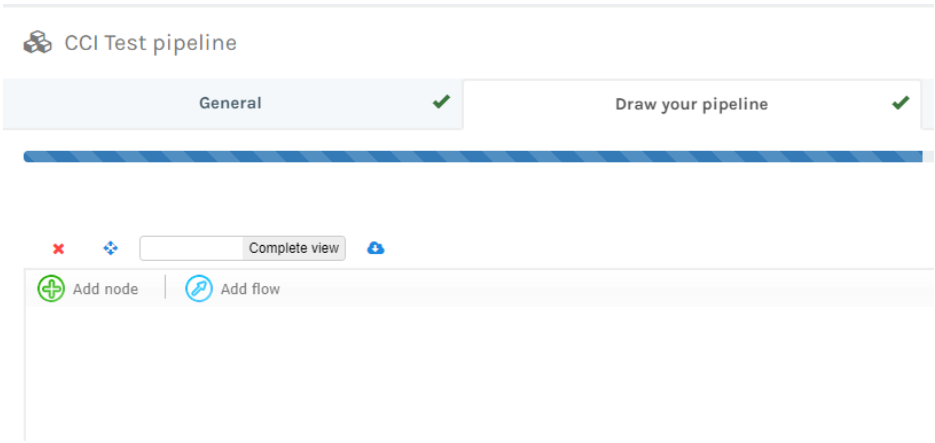
TEST-1.4.2: Pipeline execution monitoring using Max-ICS logs
Initial conditions: 1. The Service API is up and running 2. The pipeline is executing at least one job 3. The user is logged in Max-ICS and he is a contributor of the pipeline
Test execution procedure: 1. The user access to the pipeline panel  2. The user clicks access to the pipeline monitoring panel 3. The user clicks on “Logs” tab 4. The user waits that all the nodes of the pipeline log their behavior for the running job ID

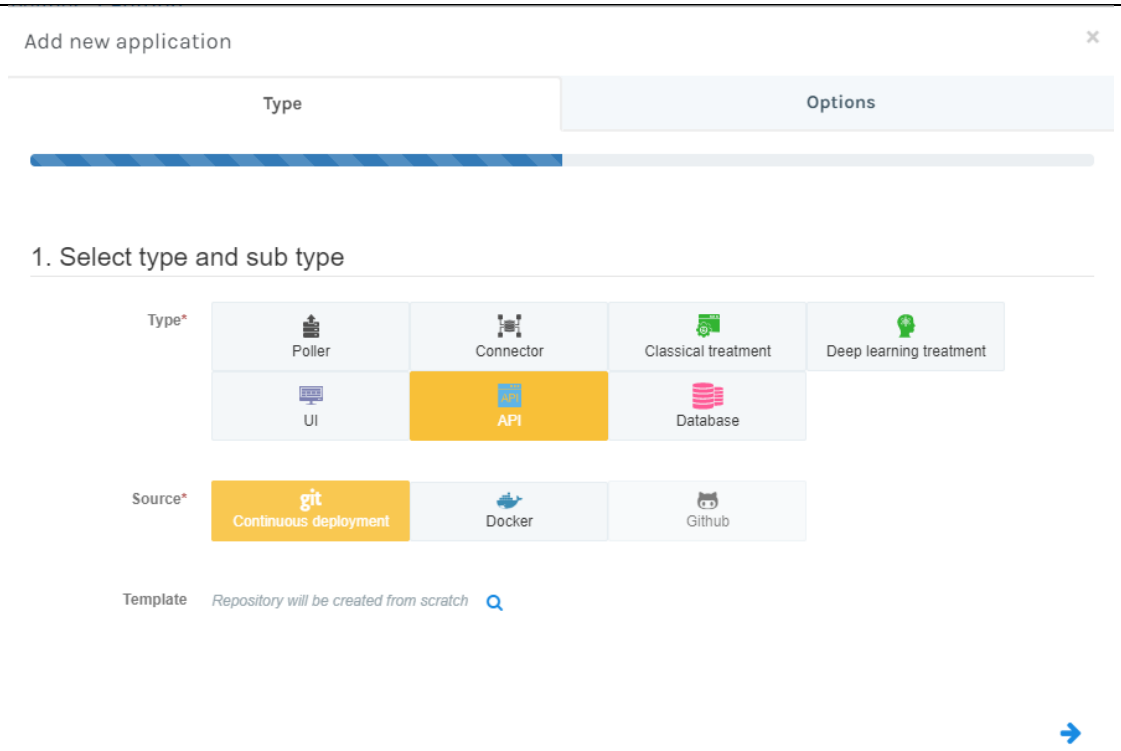
Test result:

The request is executed correctly as shown in the following screenshot:

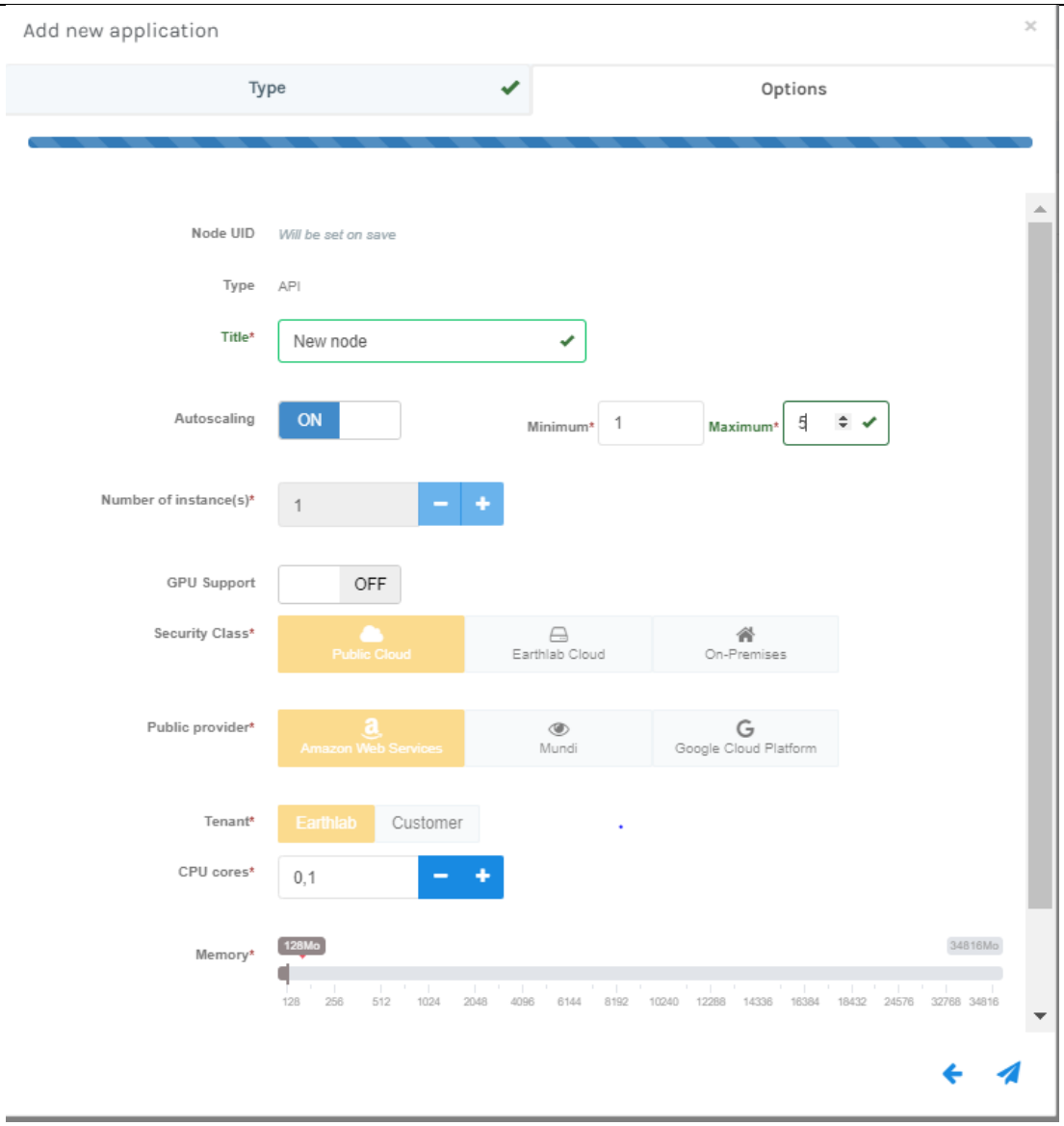


3.3.6 TEST-2.1: Pipeline Design

TEST-2.1: Pipeline Design
Initial conditions: 1. User logged in to Max-ICS 2. User created an empty workspace to design the pipeline on Max-ICS UI
Test execution procedure: 1. The user creates a node of type API by clicking “Add node”  2. The user selects API as the type of the node

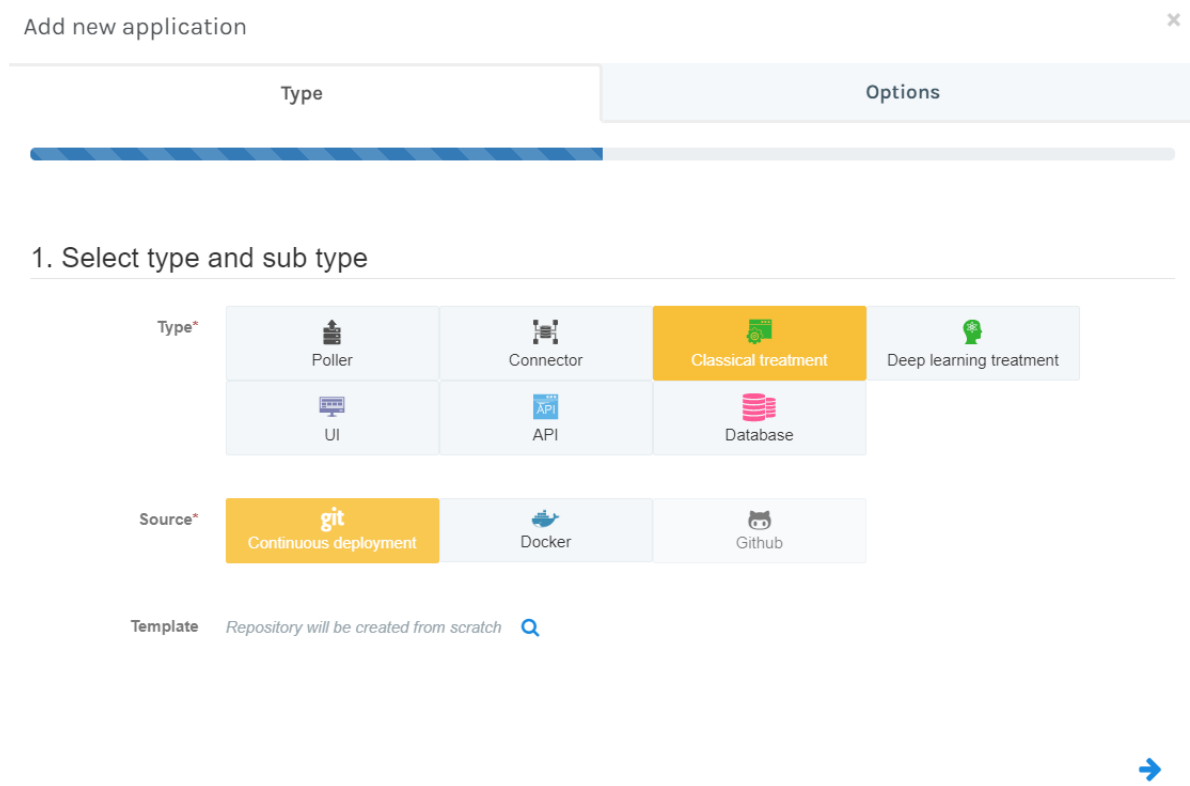


3. The user sets the node parameters (CPU, Memory, Autoscaling)

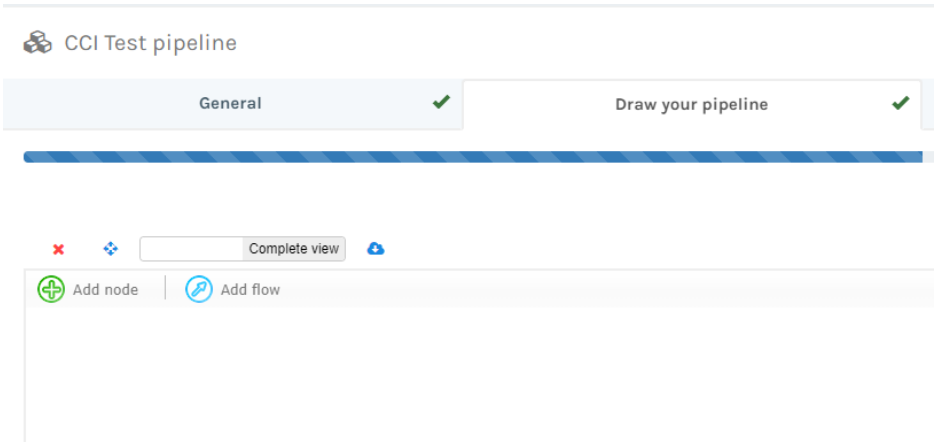


4. The user saves the node by clicking the save button

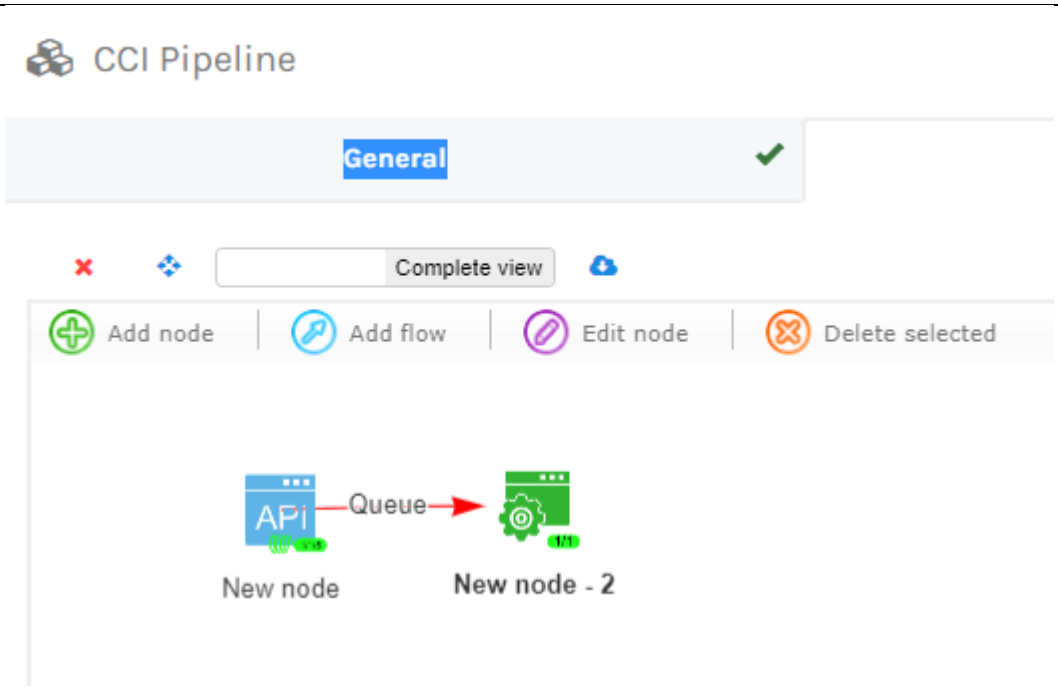
5. The user creates a node of type Classical Treatment



6. The user sets the node parameters (CPU, Memory, Autoscaling)
7. The user saves the node by clicking the save button
8. The user creates a “flow” object by clicking “Add flow”

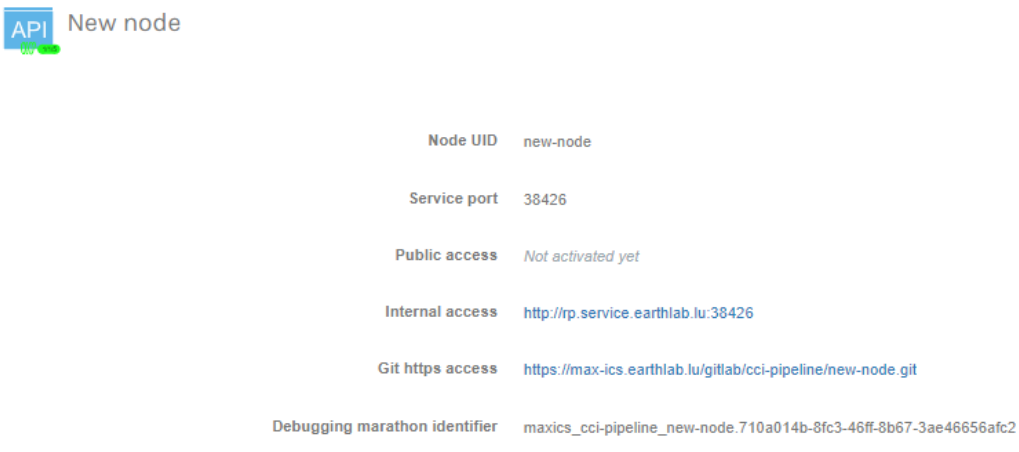


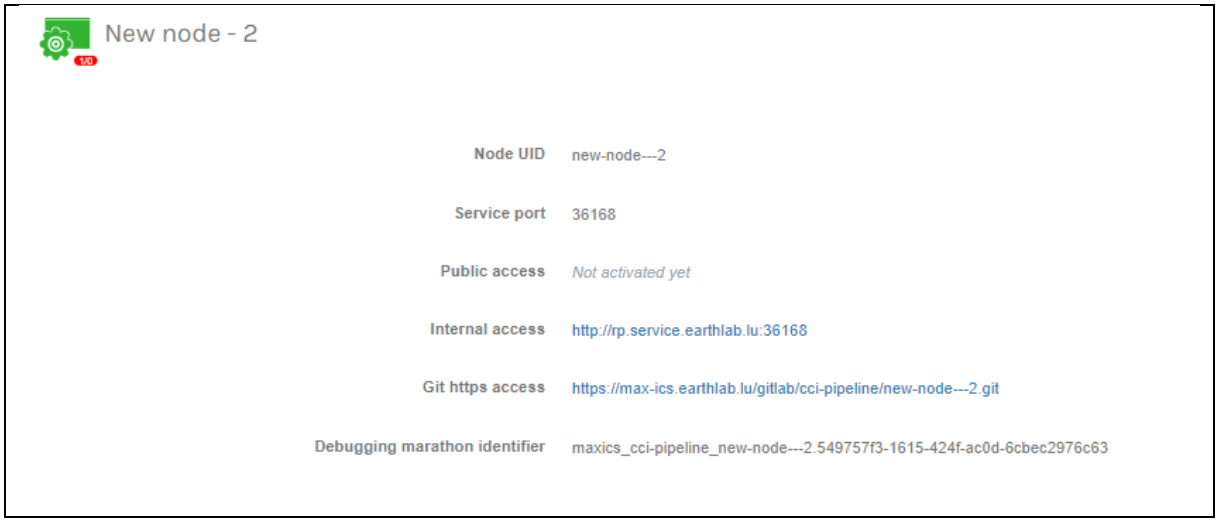
9. The user links the nodes together by using a “flow” object



Test result:

1. A git repository is automatically created for “New node”
2. A git repository is automatically created for “New node -2”
3. The “New node” is registered into a Service Registry alongside with its output queue
4. The “New node -2” is registered into a Service Registry alongside with its input queue
5. A “New node” empty instance is deployed on the infrastructure
6. A “New node-2” empty instance is deployed on the infrastructure





3.3.7 TEST-2.2: Pipeline Run

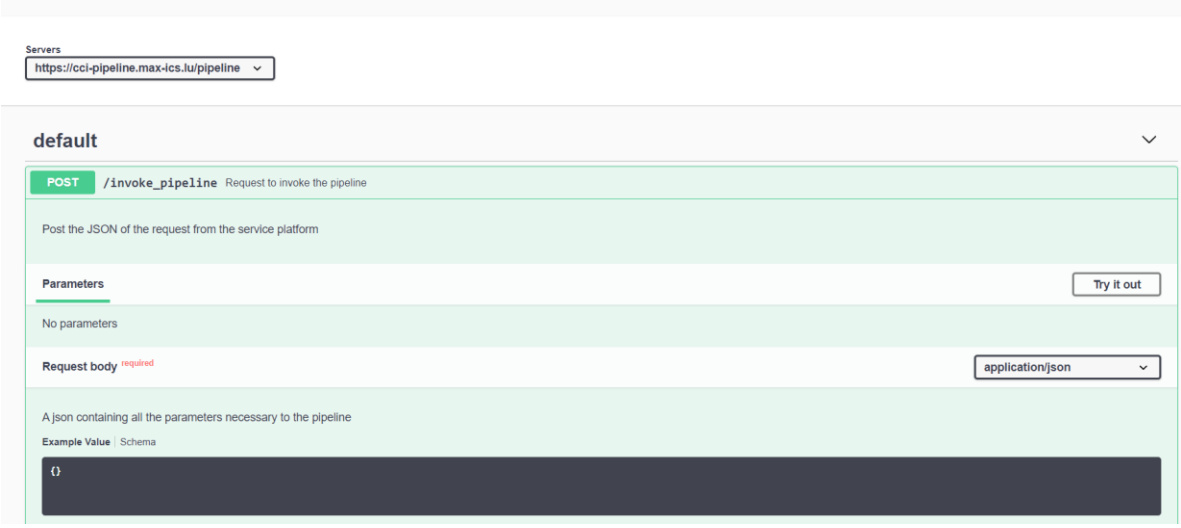
TEST-2.2: Pipeline Run

Initial conditions:

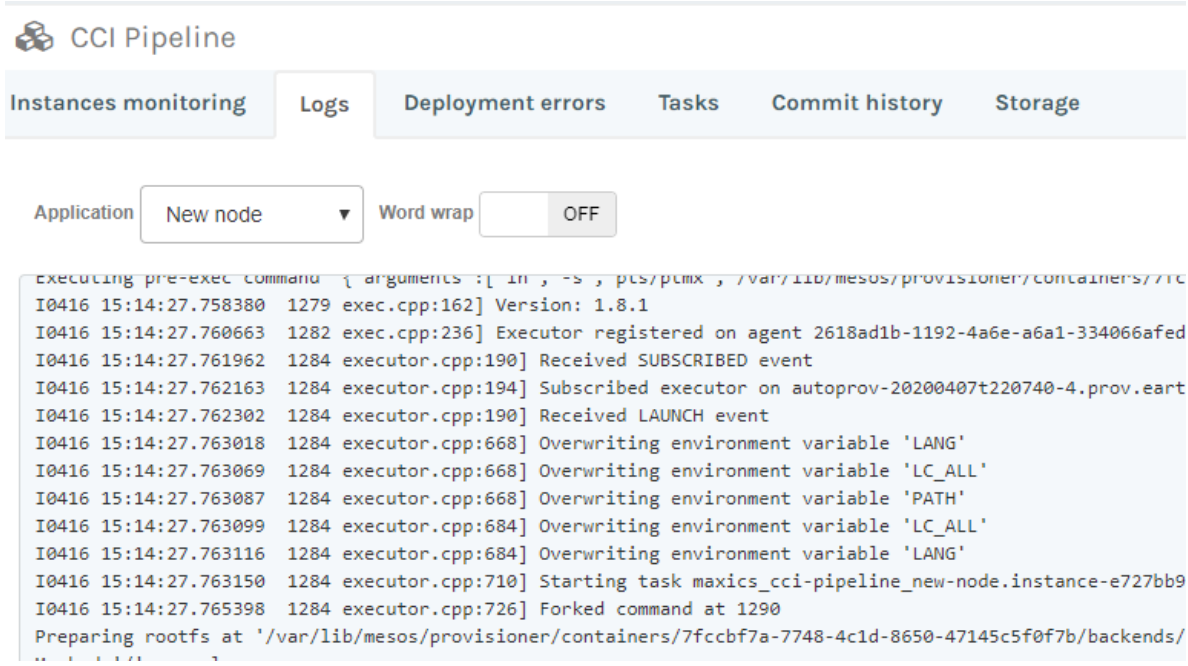
1. The user has created a pipeline on the Max-ICS PaaS
2. The user has pushed the code to each node of the pipeline to handle requests
3. The nodes code includes logs of the process status to the stdout
4. The pipeline with its nodes is up and running and reachable through internet

Test execution procedure:

1. The user sends a request to the pipeline API node with one of the following utility:
1. Curl
2. Postman
3. Python requests
4. Swagger Test UI



2. The user access the Max-ICS logs panel



Test result:

The logs show that the request has been handled

```
ents":["ln","-s","/sys/fs/cgroup/cpu,cpuacct","/var/lib/mesos/provisioner/containers/7fccbf7a-7748-4c1d-8
ents":["ln","-s","/sys/fs/cgroup/cpu,cpuacct","/var/lib/mesos/provisioner/containers/7fccbf7a-7748-4c1d-8
visioner/containers/7fccbf7a-7748-4c1d-8650-47145c5f0f7b/backends/overlay/rootfses/9a612527-5704-4bb5-a7c
[New node] ::ffff:10.99.60.14 - - [2020-04-16 15:22:11] "GET /pipeline/ui HTTP/1.1" 308 481 0.001598
15:22:11] "GET /pipeline/ui/ HTTP/1.1" 200 1557 0.004643
15:22:11] "GET /pipeline/ui/swagger-ui.css HTTP/1.1" 200 142144 0.050953
15:22:11] "GET /pipeline/ui/swagger-ui-standalone-preset.js HTTP/1.1" 200 307340 0.110798
15:22:11] "GET /pipeline/ui/swagger-ui-bundle.js HTTP/1.1" 200 974342 0.127025
15:22:13] "GET /pipeline/openapi.json HTTP/1.1" 200 852 0.000539
15:22:13] "GET /pipeline/ui/favicon-32x32.png HTTP/1.1" 200 913 0.000743
[New node] ::ffff:10.99.60.14 - - [2020-04-16 15:22:28] "GET /pipeline/ui/ HTTP/1.1" 200 1557 0.000518
15:22:29] "GET /pipeline/ui/swagger-ui-standalone-preset.js HTTP/1.1" 200 307340 0.076710
15:22:29] "GET /pipeline/ui/swagger-ui.css HTTP/1.1" 200 142144 0.076529
15:22:29] "GET /pipeline/ui/swagger-ui-bundle.js HTTP/1.1" 200 974342 0.157146
15:22:29] "GET /pipeline/openapi.json HTTP/1.1" 200 852 0.000502
15:22:30] "GET /pipeline/ui/favicon-32x32.png HTTP/1.1" 200 913 0.000786
```

3.3.8 TEST-3-1 Metadata Display

TEST-3.1: Metadata Display

Initial conditions:

1. The user has entered in the Web Interface with a user/password

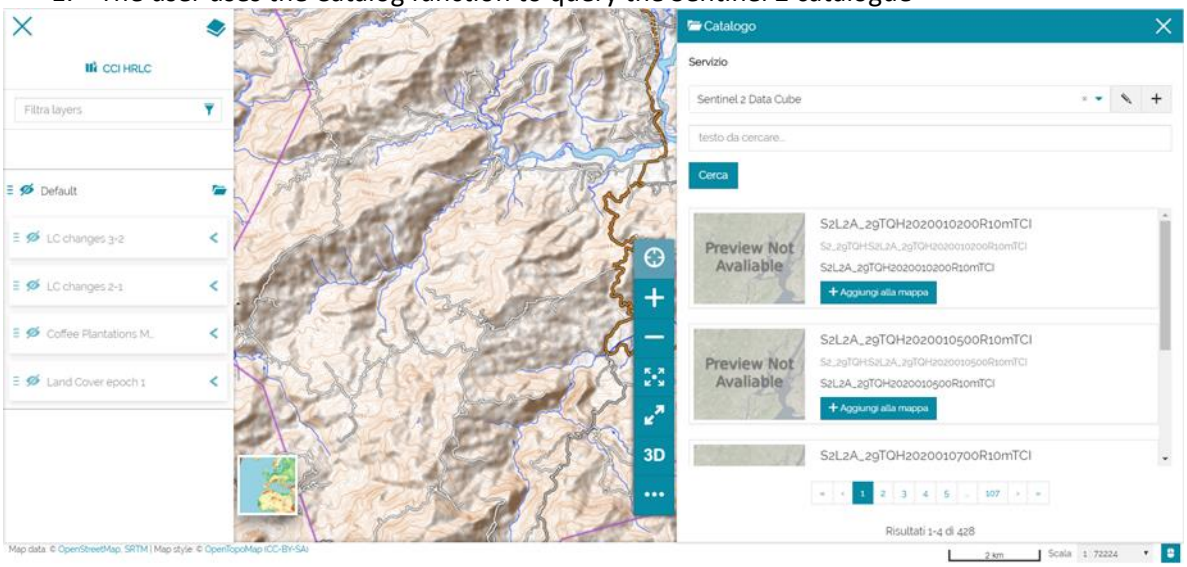
Test execution procedure:

1. The user selects the map or create a new map

2. The user uses the top right tool to add a WMS Land Cover layer from the catalogue by selecting the WMS “Sentinel 2 DataCube”
3. The user load on the map a WMS of a sentinel scene (True Color Image) of choice and a second image for the same Tile
4. The user can visualize the images on the map

Test execution procedure:

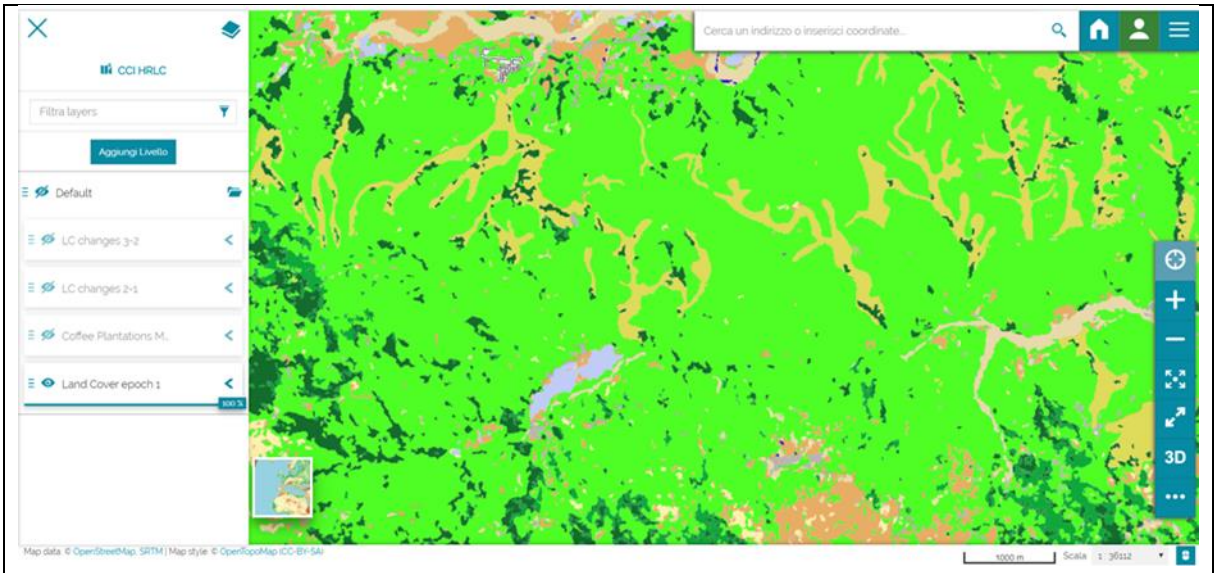
1. The user uses the Catalog function to query the Sentinel 2 catalogue



2. The user can select multiple dates from a Sentinel 2 tile and add them to the map to check for original data. In this case, in the following screenshot, an example of True Color images is shown
3.

3.3.9 TEST-3-2 Products Display

TEST-3.2: Product Display
Initial conditions: 2. The user has entered in the Web Interface with a user/password
Test execution procedure: 1. The user selects the map or create a new map 2. The user uses the top right tool to add a WMS Land Cover layer from the catalogue by selecting the WMS “CCI HRLC Test” 3. The user load on the map a WMS Land Cover layer 4. The user load on the map a WMS Change Map layer
Test result: 1. The following screenshot show an example of land-cover classification using LCCS based legend



2. The following screenshot show an example of transition map used to visualise different transition of interest