

## Horizon 2020

### H2020 LC-SPACE-04-EO-2019-2020

Copernicus Evolution – Research for harmonised and Transitional-water Observation (CERTO)

**Project Number: 870349**

Deliverable No: D7.3		Work Package: 7	
Date:	30-JUN-2021	Contract delivery due date	30-JUN-2021
Title:	First release of service prototype and documentation		
Lead Partner for Deliverable	PML Applications		
Author(s):	Oliver Clements, PMLA. Amy Westlake, PMLA. Ben Calton, PMLA		
Dissemination level (PU=public, RE=restricted, CO=confidential)			PU
Report Status (DR = Draft, FI = FINAL)			FI

#### Acknowledgements

*This project has received funding from the European Union's Horizon 2020 research and innovation programme grant agreement N° 870349*



---

## Table of Contents

1	Executive Summary .....	4
2	Introduction .....	5
3	Architecture.....	5
4	Prototype Development.....	6
5	Implementation on DIAS .....	7
5.1	Data provisioning.....	7
5.2	Workspace Creator.....	7
5.3	SNAP .....	8
5.4	Containers .....	8
5.5	CREODIAS example .....	8
6	Next Steps for Future Releases .....	10

## Reference Documents

[RD1]	CERTO Deliverable: D2.2 Technical requirements of the CERTO prototype
[RD2]	CERTO Deliverable: D7.1 Initial Report on DIAS & Cloud Provider Capabilities
[RD3]	CERTO Deliverable: D7.2 Solution Architecture Design Document

## Glossary

API	Application Programming Interface
C3S	Copernicus Climate Change Service
CMEMS	Copernicus Marine Environment Monitoring Service
CLMS	Copernicus Land Monitoring Service
CLI	Command Line Interface
DIAS	Data and Information Access Services
EO	Earth observation
FTP	File Transfer Protocol
GUI	Graphical User Interface
HPC	High Performance Computing
OGC	Open Geospatial Consortium
OpenDAP	Open-source Project for a Network Data Access Protocol
SaaS	Software as a Service
SNAP	The Sentinel Application Platform
VM	Virtual Machine
WCS	Web Coverage Service
WMS	Web Map Service

## 1 Executive Summary

Water quality is a key worldwide issue relevant to human food consumption and production, industry, nature, and recreation. The European Copernicus programme includes satellite sensors designed to observe water quality and services to provide data and information to end-users in industry, policy, monitoring agencies, and science. However, water-quality data production is split across three services, Copernicus Marine, Copernicus Climate Change, and Copernicus Land, with different methods and approaches used and some areas, notably transitional waters, are not supported by any service.

The CERTO project aims to address this lack of harmonisation by undertaking research and development necessary to produce harmonised water-quality data from each service and extend Copernicus to the large number of stakeholders operating in transitional waters. The main output of the project will be a prototype system that can be “plugged into” the existing Copernicus services, developing Data and Information Access Services (DIAS), or popular open-source software used widely by the community (e.g. SNAP).

This document outlines the first deployment of the CERTO architecture, defined in [RD3], on a DIAS platform. We also include the documentation on the setup of the system.

## 2 Introduction

This document outlines the first deployment of the CERTO processing architecture designed and documented in D7.2 [RD3]. The main purpose of this first deployment is to test that the architecture can be deployed to existing public and commercial cloud environments. The first deployment has used the CREODIAS Copernicus DIAS platform. CREODIAS provides access to a range of fully customisable virtual machines, meaning it is possible to provision complete servers which can be used in much the same way as a physical server.

This document also acts as guidance for the deployment of the system by detailing the steps necessary to deploy and configure the system. This document will continue to evolve over the lifetime of the CERTO prototype development, taking feedback from project partners and relevant technical committee members.

The CERTO developed processing granules are not currently available due to a delay created by the Covid-19 regulations limiting in-situ data collection. Due to the approach of utilising containers to encapsulate code and software dependencies, it will be relatively simple to include and execute new processing algorithms and techniques as and when they are ready.

## 3 Architecture

The system design displayed in **Error! Reference source not found.** outlines the software and tools used for the processing environment. It is arranged in three layers; layer 1 interacts with the compute providers infrastructure, layer 2 provides the compute environment and management tools, and layer 3 provides the tools to process data and deliver CERTO products.

The system architecture is designed to be agnostic of the computing infrastructure on which it will operate, allowing the user to deploy it to any infrastructure which meets the requirements. However, each infrastructure provider has implemented different methods for provisioning compute and network resources within their infrastructure; for this reason, layer 1 of the diagram is specific to the cloud hosting provider and the API they provide. These will be different between the DIAS providers, however, once the API call has been made to provision the compute resources, the rest of the setup is fully automated and provider agnostic.

Layer 2 provides the computing environment and the management tools required to request and schedule processing tasks. The perpetual controller machine is always active to receive data processing requests at any time, it then relies on Cylc (a workflow management tool) to manage the data processing tasks and Slurm (a workload scheduler) to manage where these tasks are run and if they require an extra virtual machine. Cylc and Slurm work together to handle the entire process from the initial data request through to alerting the user when the final products are available.

The applications defined in layer 3 of the diagram are a combination of existing open-source tools and CERTO specific tools and algorithms. It is these tools which allow the operator to define which products are generated.

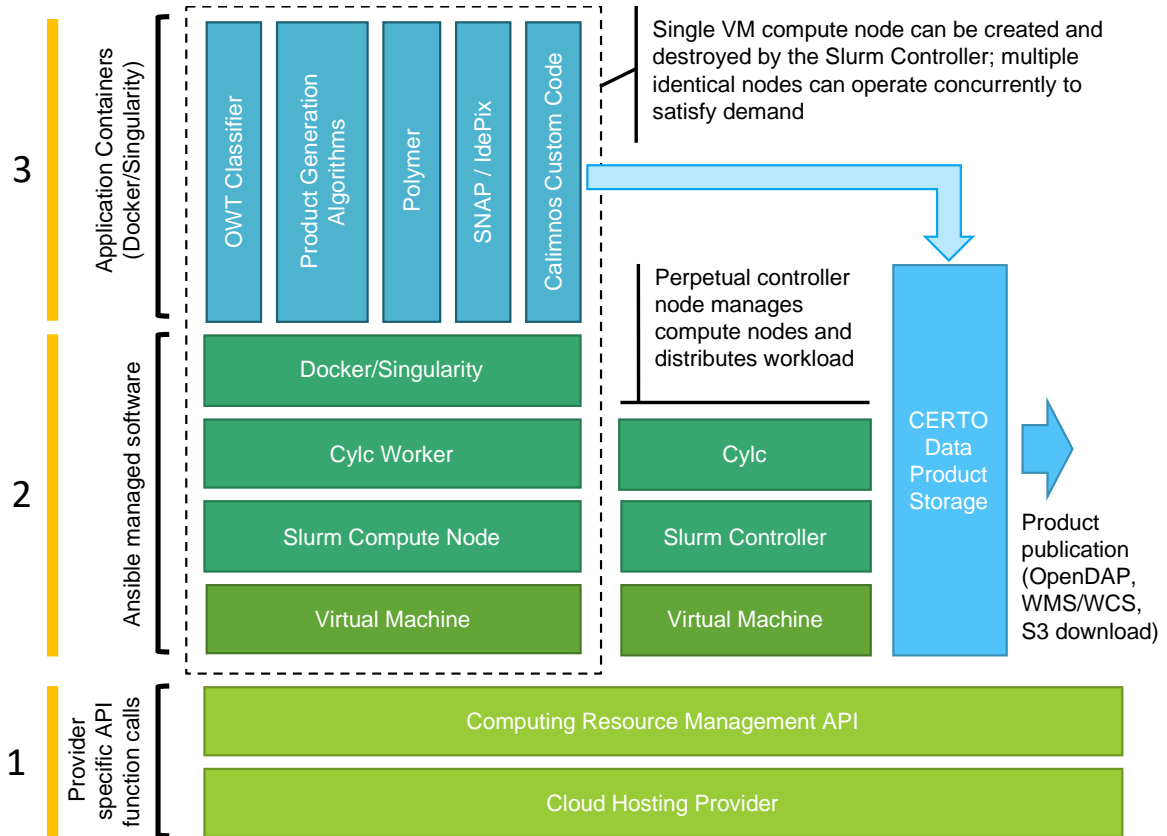












Figure 1 System architecture overview

## 4 Prototype Development

Table 1 demonstrates the development status of each of the key elements of the overall solution; a detailed explanation of the elements and the role within the prototype is available in the solution architecture design document [RD3].

Table 1 - Prototype elements and their development status

Element	Development Status	Comment
Cloud hosting API interaction	<span style="color: yellow;">■</span> In progress	CREODIAS API interactions are complete; other providers still to do

Element	Development Status	Comment
Ansible	 Complete	Ansible playbooks to install all dependent software in compute infrastructure complete and tested
Docker	 Complete	Installed via Ansible playbook
Cylc	 Complete	Installed via Ansible playbook; configured and light testing complete
Rose Suite	 Complete	Installed via Ansible playbook; example configurations can be run
Slurm	 Complete	Installed via Ansible playbook; configuration and testing complete
Catalogue search	 In progress	Catalogue search via CREODIAS API complete; other providers' catalogue search still to do
SNAP	 Complete	A containerised version of SNAP is available and tested
Polymer	 In progress	A containerised version of SNAP is available but still requires further development and testing
CERTO algorithms	 Not implemented	Not available at the time of release
Output product access	 In progress	Resulting output products are available to users in the original workspace; further developments are required to make products available via OGC services or direct download

## 5 Implementation on DIAS

### 5.1 Data provisioning

The DIAS providers offer two ways of accessing input satellite data from their API; using a GUI explorer in the browser or through a command line interface. For this initial release, the CERTO prototype will query the API through command line using customisable variables set by the user.

When data is returned through the API, each product returns a URL for downloading the data in a compressed zip format. The software will download each zip file to a newly created workspace and then unzip the files to prepare them for running through the Graph Processing Tool (gpt) tool for processing, saving all the finished processed files on the same workspace.

### 5.2 Workspace Creator

A workspace is an allocation of storage where all input data (satellite data, ancillary data, configurations) and intermediate products generated by tasks in the processing chain are stored, and is a space where log files record activity. The workspace creator defines this storage ensuring that all required inputs are present as a prerequisite to processing starting.

These workspaces are created and destroyed as and when needed by Slurm, the workload manager.

### **5.3 SNAP**

SNAP is a Java based application developed by Brockmann Consult which allows users to interact with and manipulate satellite data. The CERTO prototype will be making use of SNAP data processing features through its Graph Processing Tool (gpt), a command line tool for automating and performing SNAP data manipulations on multiple data files. SNAP and gpt will be run within their own container keeping the data processing separate from the rest of the application.

### **5.4 Containers**

The use of containers within the prototype ensures that code and software libraries are kept separate from the compute environment they are installed on; this enables the container to be replicated as many times as necessary, makes troubleshooting easier, and means that when the CERTO specific developments (improved algorithms, atmospheric correction, etc.) are ready they can be inserted as an add-on to the existing software.

### **5.5 CREODIAS example**

With the use of containers, the prototype is expected to be able to run on any of the main DIAS services; for development and testing, we have chosen to use CREODIAS.

Referencing the diagram in section 3 (Fig 1), we have been focussing on the development of the perpetual controller that will accept user arguments and return the data to be downloaded.

Within this controller we have the main docker file which is based on ubuntu:latest and runs primarily Python3. The default command for this container is to run the app.py file, returning a html page with a form which requests details about the data the user wants to request. On submission of the form, the variables chosen by the user are placed into the api call to CREODIAS which returns urls to download data.

The current processing flow for the docker container is below.



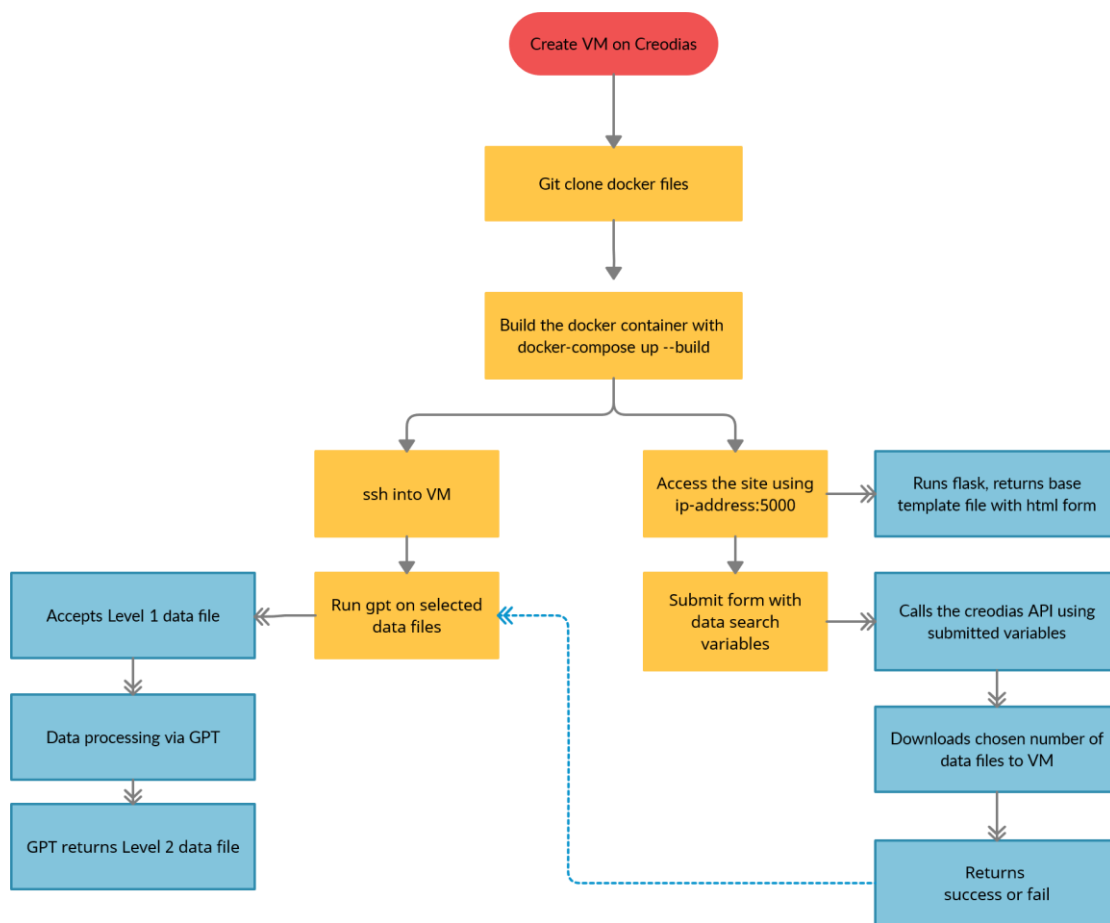


Figure 2 Flow diagram showing the route to execution

It is expected that the front end Python/flask app may not be provided in the final product, however, for testing and development purposes it is currently being included. Most, if not all, of the processing is expected to be carried out programmatically without the need for a front-end interface.

## 5.4 Installation Guidance

Any of the DIAS services that provide virtual computing environments can run the prototype, all that is initially required is a user account of the chosen DIAS.

To get the prototype running, two virtual machines need to be created on the DIAS; a Slurm worker and a Slurm controller. Both virtual machines need to be running Ubuntu 20.04 and have a minimum of 1GB ram. The Slurm controller will be the main machine where processes are run from, so users will need to ssh into their controller machine and generate a keypair to use with the worker machines.

If viewing access to the controller is required, a floating IP will need to be created and associated with the controller machine with viewing access set on port 80.

Next, users will need to ssh onto the controller machine and install ansible, clone the git repository and run the `initial_setup.sh` file. Running this file successfully will install python and all the other dependencies required by this project.

Once the controller machine is set up, the docker container needs to be downloaded from git and then built using the “`docker-compose up --build`” command which spins up the first container and downloads all the required software. A CREODIAS account and keycloak token is required for the API calls to return data successfully. A CREODIAS username and password can be set in the `config.ini` file and the python code in the docker container will handle generating the keycloak token before an API call is made.

Within the container the docker file runs `ubuntu:latest` and primarily uses Python3. The default command for this container is to run the `app.py` file, returning a html page with a form which requests details about the data the user wants to request. On submission of the form, the variables chosen by the user are placed into an API call to CREODIAS which returns urls to download data.

## 6 Next Steps for Future Releases

This document highlights that the proposed CERTO processing architecture can be deployed on an existing Copernicus DIAS platform. This, similarly, shows that it would be able to be deployed to commercial cloud environments such as AWS however these cloud systems wouldn't provide “local” access to the relevant datasets for the CERTO project. Through the use of containers we are able to deploy novel processing code with relative ease, making the system flexible and adaptable to advancements made within the CERTO project.

Over the coming months the CERTO team will be producing processing code and the system as defined and installed above will allow rapid deployment as and when it becomes available. This will prove beneficial through the iterative development process which processing algorithms undergo.