OGC Earth Observation Applications Pilot

CRIM Engineering Report

Publication Date: 2020-10-26

Approval Date: 2020-09-23

Submission Date: 2020-08-27

Reference number of this document: OGC 20-045

Reference URL for this document: http://www.opengis.net/doc/PER/EOAppsPilot-CRIM

Category: OGC Public Engineering Report

Editor: Tom Landry

Title: OGC Earth Observation Applications Pilot: CRIM Engineering Report

OGC Public Engineering Report

Table of Contents

Chapter 1. Subject

This engineering report documents experiments conducted by CRIM in OGC's Earth Observation Applications Pilot project, sponsored by the European Space Agency (ESA) and Natural Resources Canada (NRCan), with support from Telespazio VEGA UK. Remote sensing, machine learning and climate informatics applications were reused, adapted and matured in a common architecture. These applications were deployed in a number of interoperable data and processing platforms hosted in three Canadian provinces, in Europe and in the United States.

Chapter 2. Executive Summary

CRIM’s key findings are as follows:

- From the application developer’s perspective, it was easier to first develop and test locally, and then use an OGC Application Programming Interface (API) to deploy and execute remotely. This is partly due to a smoother learning curve from application to packaging, than from package to platform.
- Conformance classes for an Application Deployment and Execution Service (ADES) API should be standardized. An offset in the support of API elements subsists in participants' implementations.
- Application packaging, deployment and execution are appropriately defined by combination of Common Workflow Language (CWL) and Docker images.
- More tests are required to better support application workflows running in multiple platforms. In order to build a federated cloud, these tests have to run regularly, in a structured and systemic fashion.
- Participants' findings should offer feedback into EOEPCA, for example for platforms intending to support machine learning (ML) services. From the application developer’s perspective, ML apps are well defined and should be deployable like any other app. It is not clear with the current EOEPCA use cases that the platforms will easily integrate ML services (annotations, trained models, access to GPU clusters, etc.).

The business value of this initiative is articulated in CRIM’s promotional videos, expressed in both the perspective of platform [https://www.youtube.com/watch?v=QkdDFGEfIAY] managers and application [https://www.youtube.com/watch?v=no3REyoxE38] developers. The figure below presents the key value propositions of multidisciplinary workflows and federated infrastructures.

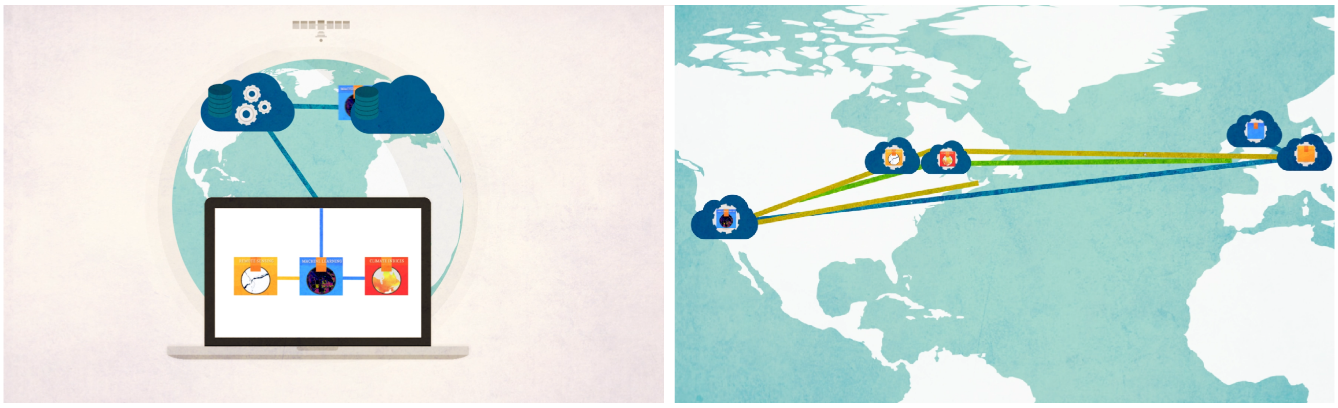


Figure 1. Still frames of CRIM’s platform and application promotional videos

Again, the reader is invited to consult the associated promotional videos for CRIM’s research and technological transfer motivations. Additionally, the following elements are noted:

- See Section 6 for motivations related to the Earth Observation Data Management System (EODMS) and Pacific Boreal Cloud (PBC)
- Machine learning applications, tools and services are of a particular interest with respect to modern architectures such as EOEPCA.

- There is a need for a base, common stable applications package shared with the community, for example the Sentinel Application Platform (SNAP).

As summary of recommendations, and to introduce potential future work, CRIM proposes the following:

- Establish a clear intention and roadmap with respect to CWL
  - Seek stronger links with CWL community. Ex: IPython2CWL [https://pypi.org/project/ipython2cwl/]
  - Consider de facto standardization of CWL for *ProcessDescription* and AP
- Increase test coverage, depth and capabilities related to the common architecture
  - Consider expanding and enforcing an OGC test suite.
  - Provide additional examples, demonstrations, documentation and tutorials to application developers.
  - Describe and normalize hardware requirements of applications in the AP, and inversely, expose platform capabilities to application users and developers.
  - Establish base tests for applications, but also for system integrity of a federated cloud.
  - Test the SNAP Application Package (AP) with various data and I/O with GDAL, or other mission-specific modules or operators.
  - Provide additional QA of Rardarsat-1 and Radarsat Constellation Mission (RCM) read-writes operators, and integrate into SNAP. Mark RCM or RS-1 files with defective metadata in the catalogues.
- Seek new projects and initiatives to continue maturation of the architecture and the infrastructure
  - Consider another pilot to develop federated workflows and test, for example cumulative effect project. For example, remote sensing applications (feature production, band management, ARD, datacubes) that can feed into machine learning application (detectors, classifiers).
  - Consider global, national and regional analysis applications combining observational outputs with long-term climate projections and indices.
  - Consider mapping the cumulative effect workflows and applications onto SDGs targets.

2.1. Document Contributor Contact Points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Tom Landry	CRIM	Editor
Francis Charette-Migneault	CRIM	Contributor

Name	Organization	Role
Mario Beaulieu	CRIM	Contributor
Mathieu Provencher	CRIM	Contributor
Louis-David Perron	CRIM	Contributor
David Byrns	CRIM	Contributor
Samuel Foucher	CRIM	Contributor
William Mackinnon	NRCAN	Contributor
Ryan Ahola	NRCAN	Contributor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2], 2010
- OGC: OGC: OGC 13-026r8, OGC OpenSearch Extension for Earth Observation 1.0 [https://portal.opengeospatial.org/files/13-026r8], 2016
- OGC: OGC: OGC 14-065r2, OGC Web Processing Service 2.0.2 Interface Standard Corrigendum [https://portal.opengeospatial.org/files/14-065r2], 2018
- OGC: OGC: OGC 13-032r8, OGC OpenSearch Geo and Time Extensions 1.0.0 [https://portal.opengeospatial.org/files/?artifact_id=56866], 2014
- CWL: CWL group: Common Workflow Language Specifications, v1.1 [https://www.commonwl.org/v1.1/], 2020

Additionally, the following unpublished document is referenced in this document.

- OGC: OGC: 18-062, OGC API - Processes - Part 1: Core candidate standard [https://www.ogc.org/standards/requests/216]

Chapter 4. Terms and Definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- Climate Index

  ```
  A term used to refer to properties of the climate that are not measured in the
  field or calculated by climate models but rather that are calculated or derived
  from climate variables such as temperature and precipitation. Examples include the
  number of growing degree-days, freeze-thaw cycles, and the drought code index (see
  variable).
  ```

- Climate Information

  ```
  Refer to climatic data that describe either past conditions, obtained from
  meteorological observations (stations, satellites, radars), or the future, obtained
  from the outputs of climate models.
  ```

- Climate Model

  ```
  A numerical representation of the climate system based on the physical, chemical,
  and biological properties of its components, their interactions and feedback
  processes, and accounting for most of its known properties.
  ```

- Climate Variable

  ```
  The term climate variable is used to refer to a variable that can be measured
  directly in the field (at meteorological stations for example) or that is
  calculated by climate models. (See Index)
  ```

- CMIP5

  ```
  Coupled Model Intercomparison Project, Phase 5. CMIP5 is a coordinated climate
  modeling exercise involving 20 climate-modeling groups from around the world. It
  has provided a standard experimental protocol for producing and studying the output
  of many different global climate models. The output from CMIP5 ensemble experiments
  is used to inform international climate assessment reports, such as those from the
  IPCC.
  ```

- Downscaling

```
A method that can provide climate model outputs at a finer resolution than their
original resolution. Two different approaches are prioritized: statistical
downscaling and dynamic downscaling.
```

- Statistical Downscaling

```
This type of downscaling relies on the use of statistical relationship that relates
large scale climate features, named predictors, to predictants like local climate
variables. (See downscaling)
```

4.1. Abbreviated Terms

- ACL - Access Control List
- ADES - Application Deployment and Execution System
- AOI - Area Of Interest
- AP - Application Package
- API - Application Programming Interface
- ARD Analysis Ready Data
- CCCS - Canadian Center for Climate Services
- CF - Climate Forecast
- CNN - Convolutional Neural Network
- CGDI - Canadian Geospatial Data Infrastructure
- CRIM - Computer Research Institute of Montreal
- CSW - Catalogue Service for the Web
- CWL - Common Workflow Language
- DKRZ - Deutsches Klimarechenzentrum
- DL - Deep Learning
- DNN - Deep Neural Network
- ECCC - Environment and Climate Change Canada
- ER - Engineering Report
- EMS - Execution Management System
- EO - Earth Observation
- EOC - Earth Observation Clouds
- EODMS - Earth Observation Data Management System
- EOEPCA - Earth Observation Exploitation Platform Common Architecture
- ESA - European Space Agency

- ESGF - Earth System Grid Federation
- EP - Exploitation Platform
- GDAL - Geospatial Data Abstraction Library
- GFCS - Global Framework for Climate Services
- GPT Graph Processing Tool
- IdP - Identity Provider
- IPCC - Intergovernmental Panel on Climate Change
- JSON - JavaScript Object Notation
- LLNL - Lawrence Livermore National Laboratory
- ML - Machine Learning
- NRCan - Natural Resources Canada
- OGC - Open Geospatial Consortium
- OWS - OGC Web Services
- QA - Questions and Answers
- REST - Representational State Transfer
- SDG - (United Nations) Sustainable Development Goals
- SNAP - Sentinel Application Platform toolbox
- TB - Testbed
- TEP - Thematic Exploitation Platform
- TIE - Technology Integration Experiments
- TOI - Time Of Interest
- URI - Uniform Resource Identifier
- URL - Uniform Resource Locator
- VM - Virtual Machine
- WCS - Web Coverage Service
- WFS - Web Feature Services
- WMS - Web Map Service
- WPS - Web Processing Service
- WPS-T - Transactional Web Processing Service

Chapter 5. Overview

Section 5 introduces the Earth Observation (EO) platform architecture, components, implementations and endpoints. It also lists the main experiments conducted with all participants platforms and applications.

Section 6 presents the remote sensing application proposed by CRIM in this pilot.

Section 7 describes the machine learning application and models developed by CRIM.

Section 8 explains the climate processes interoperability experiments aiming to support climate services in Canada.

Section 9 discusses more generally about Earth Observation application packages and their relationship with the platforms.

Section 10 concludes the report by offering a summary of the findings and recommendations.

Annex A provides a list of climate indices available through Finch WPS.

Annex B provides a sample process description for computation of climate indices in Finch WPS.

Annex C provides a sample remote sensing processing graph to run in the Graph Processing Tool (GPT) of SNAP.

Annex D provides a JSON file for the climate process Execute request body.

Annex E provides a CWL file to register an external WPS 1.0 provider.

Annex F provides a JSON request body to execute *WorkflowSubsetPicker*.

Annex G provides a list of all Technology Integration Experiments (TIEs) conducted by CRIM.

Annex H provides an example of configurations for S3 buckets support.

Chapter 6. Earth Observation Platform

This section describes the various EO platforms deployed and integrated in this pilot by CRIM. It presents a high-level view from the perspective of platform developers.

6.1. Overview

For several years now, CRIM has been developing research software for its own researchers, and for the larger Canadian scientific community. A key milestone in this series of initiatives is project *PAVICS* (Plateforme pour l'Analyse et la Visualisation de l'Information Climatique et Scientifique), funded by CANARIE, and aiming to facilitate the Big Data workflow of climate scientists. This cloud research platform offers tailored climate processes as OGC WPS, as well as data services such as WCS, WFS and WMS. In order to modernize the open standards on which it relies, and to enable integration of other applications and expertise at CRIM, *PAVICS* platform was reused in parts in OGC Testbed-13 to 16. The EO Applications pilot is therefore a culminating effort to mature previous prototypes towards a more operational status. Two other important initiatives spurred from *PAVICS*. Firstly, *ClimateData.ca*, a climate information portal that enables Canadians to access, visualize, and analyze climate data, currently uses *PAVICS* as an analytical backend. Secondly, project *DACCS* (Data Analytics for Canadian Climate Services), funded by the Canada Foundation for Innovation (CFI), greatly extends *PAVICS* scope. This project aims to bridge the gap between climate and EO platform by integrating remote sensing and machine learning uses cases. In the upcoming years, project *DACCS* is expected to continue development of *PAVICS* and to support ClimateData.ca climate services.

Another participant, sponsor and beneficiary of the pilot project is Natural Resources Canada (NRCan). The Canadian Forest Service (CFS) provided cloud resources through *Pacific Boreal Cloud*, but also EO data and forestry use cases to drive innovation, in this pilot and since Testbed-13 [http://docs.opengeospatial.org/per/17-035.html]. In parallel, NRCan is also investigating possible use of the architecture for its Earth Observation Data Management System (EODMS). While the architecture has shown promise within experimental testing environments, further investigation using real-world scenarios is required. Through this pilot project, NRCan evaluates the architecture on a version of EODMS, using data and applications that represent common use cases for EODMS clients. Results will highlight a path forward for improvements to the architecture to fully meet real-world operational needs, as well as defining a roadmap for EODMS's transition to an exploitation platform. It is also hoped that this architecture will improve the ability of Canada's remote sensing scientists to leverage Canadian and international EO information in a seamless manner.

6.2. EO Platforms Deployment

One for the major particularities of CRIM's contribution to the pilot is the installation of its ADES/EMS solution, *Weaver*, on three separate clouds. Additionally, a local EMS sent requests to Spacebel's ADES. The intent is to deploy and run applications on the most appropriate platform, and to allow elaboration of more advanced operational scenarios. The following figure presents the deployment diagram of platform elements.

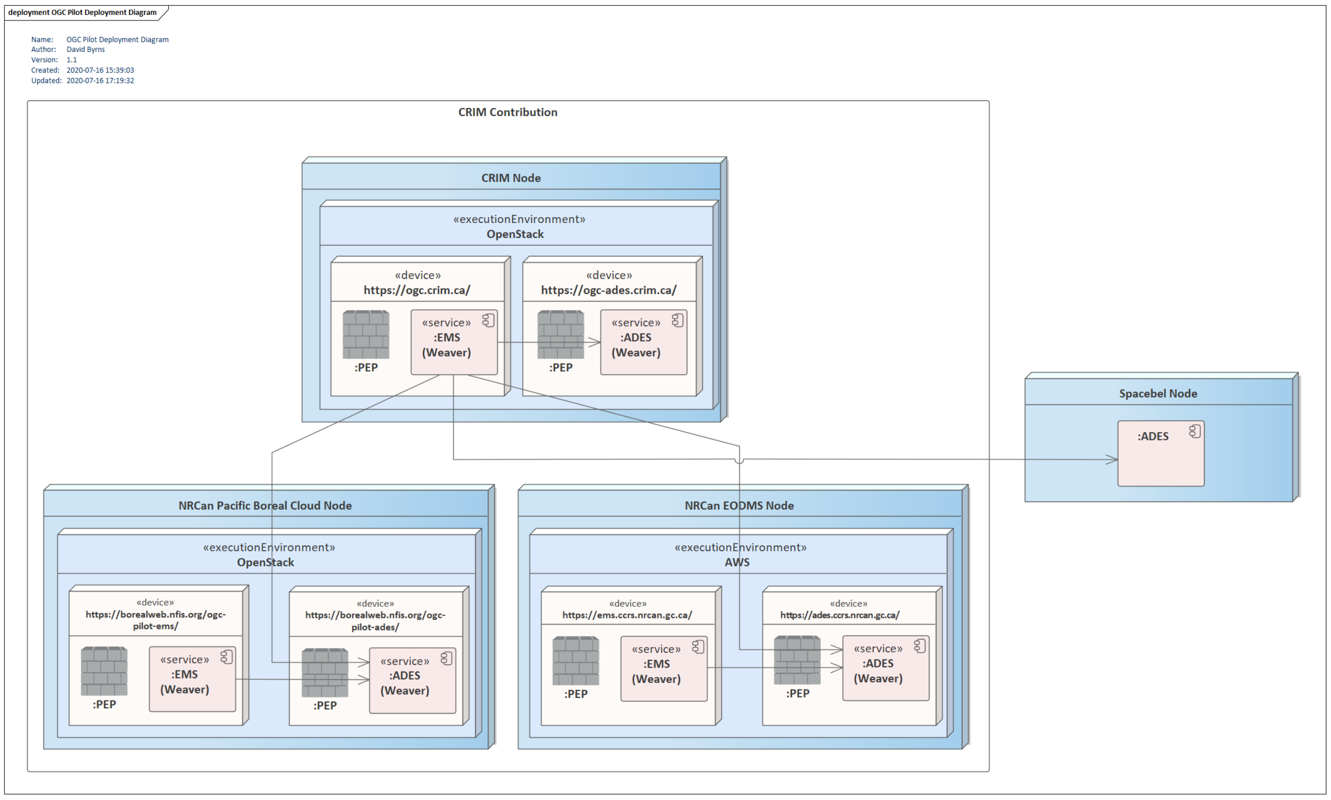

deployment OGC Pilot Deployment Diagram
Name: OGC Pilot Deployment Diagram
Author: David Byrns
Version: 1.1
Created: 2020-07-16 15:39:03
Updated: 2020-07-16 17:19:32
CRIM Contribution
CRIM Node
«executionEnvironment» OpenStack
«device» https://ogc.crim.ca/
:PEP
«service» :EMS (Weaver)
«device» https://ogc-ades.crim.ca/
:PEP
«service» :ADES (Weaver)
Spacebel Node
:ADES
NRCan Pacific Boreal Cloud Node
«executionEnvironment» OpenStack
«device» https://borealweb.nfis.org/ogc-pilot-ems/
:PEP
«service» :EMS (Weaver)
«device» https://borealweb.nfis.org/ogc-pilot-ades/
:PEP
«service» :ADES (Weaver)
NRCan EODMS Node
«executionEnvironment» AWS
«device» https://ems.ccrs.nrcan.gc.ca/
:PEP
«service» :EMS (Weaver)
«device» https://ades.ccrs.nrcan.gc.ca/
:PEP
«service» :ADES (Weaver)

Figure 2. Deployment diagram

The table below lists each of the deployments of ADES and EMS servers. The specificities of the *configurations* can be found in the appropriate subsection below.

Table 1. Endpoints

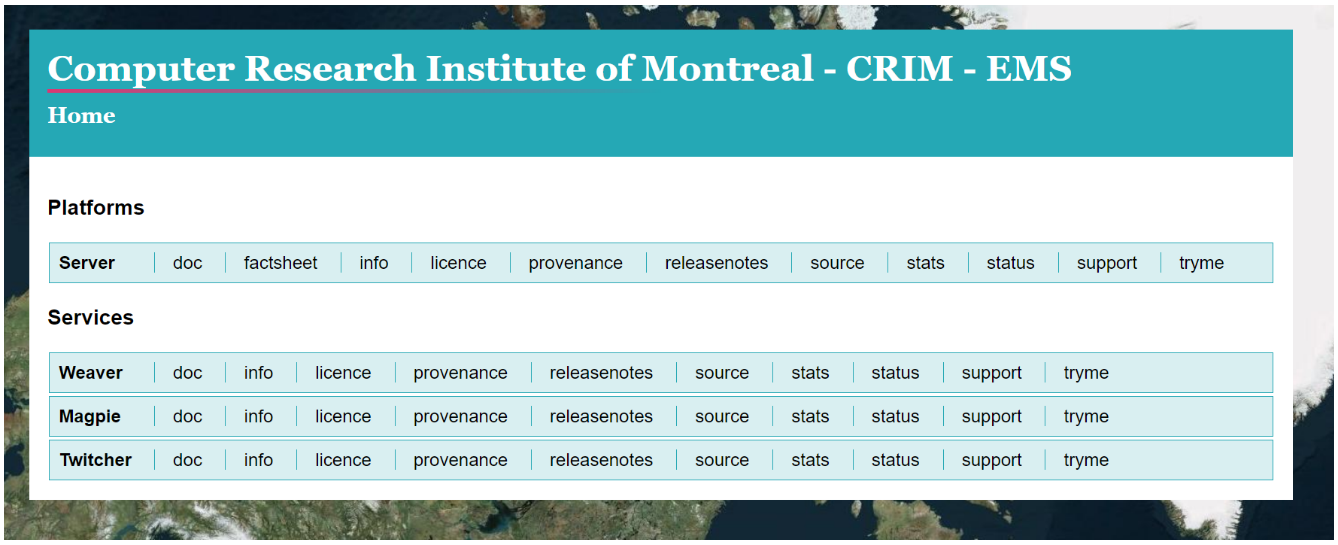
Type	Usage	Endpoint URL	Configuration	Cloud vendor
EMS	CRIM Dev	https://ogc-ems.crim.ca	Generic	OpenStack
ADES	CRIM Dev	https://ogc-ades.crim.ca	Generic	OpenStack
EMS	CRIM Prod	https://ogc.crim.ca	Finch	OpenStack
EMS	ClimateData.ca Prod	https://pavics.climatedata.ca	Finch	cloud.ca, VMWare
EMS	CRIM Demo	https://finch.crim.ca	Generic	OpenStack
ADES	NRCAN Pacific Boreal Cloud	https://borealweb.nfis.org/ogc-pilot-ades	Generic	OpenStack
EMS	NRCAN Pacific Boreal Cloud	https://borealweb.nfis.org/ogc-pilot-ems	Generic	OpenStack
ADES	NRCAN EODMS	https://ades.ccrs.nrcan.gc.ca	Generic	Amazon AWS

Type	Usage	Endpoint URL	Configuration	Cloud vendor
EMS	NRCAN EODMS	https://ems.ccrs.nrcan.gc.ca	Generic	Amazon AWS

6.2.1. API routes

The *OGC API - Processes* path is obtained by appending */ems/api* and */ades/api* to the EMS and ADES endpoint URLs, respectively, while processes can be found at */ems/processes* and */ades/processes*.

To facilitate research platform software re-use, CANARIE requires that all platforms developed under the Research Software program, like *PAVICS*, support the CANARIE Platform and Service Registry and Monitoring System [https://science.canarie.ca/researchsoftware/services/doc/RPI%20API%20Enhancements%20for%20CANARIE%20Service%20Registry%20and%20Monitoring%20System_en.html]. For each endpoint and as depicted in the next figure, the */canarie* route provides the registry and monitoring API that fully describe the details of the platform, how to use it, and how to obtain assistance. CANARIE's platform monitoring service also measures the availability and usage of research platforms.


Computer Research Institute of Montreal - CRIM - EMS

Home

Platforms

Server	doc	factsheet	info	licence	provenance	releasenotes	source	stats	status	support	tryme

Services

Weaver	doc	info	licence	provenance	releasenotes	source	stats	status	support	tryme
Magpie	doc	info	licence	provenance	releasenotes	source	stats	status	support	tryme
Twitcher	doc	info	licence	provenance	releasenotes	source	stats	status	support	tryme

Figure 3. CANARIE API of https://ogc-ems.crim.ca/canarie/

6.2.2. CRIM Hybrid Cloud

For the pilot, CRIM provided resources from its private cloud. These cloud resources have been provisioned and used continuously in Testbed-13, 14, 15 and 16. The basic provisioned hardware is composed of twelve OpenStack VM m4.large (2 VCPU, 8GB RAM, 200GB DISK) and three volumes of 2TB on spinning-disk. These resources have been extended in the pilot to accommodate applications with large memory footprints. Virtual machines are deployed in a public OpenStack tenant. Its IT department supports daily operations, while R&D personnel manages a large part of its application space. CRIM, being part of *Réseau d'informations scientifiques du Québec* (RISQ) [https://www.risq.quebec/fr/le-reseau-du-risq] and CANARIE network [https://www.canarie.ca/network/], has access to a pan-Canadian high-speed network dedicated to research.

6.2.3. Pacific Boreal Cloud (PBC)

The Canadian Forest Service (CFS) presented or supported several experiments in OGC's innovation program, in part by contributing resources on the *Pacific Boreal Cloud* (PBC), a high-performance cloud infrastructure at the Pacific Forestry Centre in Victoria. The center provided virtual machines, storage, networking, as well as EO data, to enable numerous experiments such as biomass estimation using point clouds, cloudless mosaicking, tree species recognition or lake-river discrimination.

In OGC Testbed-13, CFS expressed interest in extracting polarimetric parameters from Radarsat-2 SQW data using a combination of OGC Web services and Cloud environments. The requirements for such EO processing are well inline with the remote sensing application presented in this pilot in Section 7. In the architecture envisioned at that moment, resources are only used when necessary, thus reducing overhead costs of maintaining expensive servers or computing power. Assuming successful research and development, CFS could consider processing larger regions in Canada, and possibly for the National Forest Inventory Plots across the country. Later, in Testbed-14, CRIM assessed the feasibility of installing its EMS on the PBC, but did not proceed at the time. CRIM also accessed the PBC to ensure that application packages could be deployed by an ADES hosted on the cloud. Finally, the Distributed Access Control System (DACS) security solution was tested and briefly evaluated. For this pilot, one key expectation of Pacific Forestry Service and sponsors is the deployment of an ADES/EMS pair and successful execution of a remote sensing application.

Summary of Experiments

All previous ADES/EMS deployments were designed to have their own host name (distinct for each one), with the ADES/EMS exposed at the root URL. To expose the service externally at PBC, it was mandatory to use the proxy provided by PBC, with a common host name (borealweb.nfis.org) and an added prefix to the URL path to differentiate between the ADES/EMS residing under the shared host name (*/ogc-pilot-ades/* or */ogc-pilot-ems/*). CRIM performed code and configuration changes to adapt some of the components to support this environment. Moreover, communications between PBC's proxy and ADES/EMS deployments were in HTTP instead of HTTPS. Changes were needed to support this difference to properly redirect incoming requests to the appropriate services (*Magpie*, *Twitcher*, *Weaver*).

For applications, the stacker app (#TIE1001) was run but error in SNAP - an underlying application in deployed process package. Another application was successfully executed (#TIE1302) to validate the server configuration, and that stacker failure was only caused by the failing package dependency. In that case, the executed application converts a JSON-file with literal NetCDF links into direct HTTP references to NetCDF files which can be exposed externally.

CRIM deployed the servers twice on the PBC: once for initial deployment/configuration, and once to update Weaver to more recent version following EODMS developments. The latest deployed version on PBC for ADES, EMS and apps is now in sync with EODMS. Experimental feedback from NRCan CFS is pending regarding the execution of more advanced applications, for specific use cases, as per their desired functionalities.

6.2.4. EO Data Management System (EODMS)

NRCan is responsible for the operation of the EODMS. EODMS provides an archiving and discovery system for the Government of Canada's EO data (e.g. satellite imagery). As EODMS facilitates federal and public access to crucial geospatial information, it represents a core component of the CGDI.

New developments in the EO domain present challenges to the ongoing viability of the current EODMS architecture. Traditionally, EO users have identified appropriate imagery in repositories, downloaded the required information and finally applied necessary processing on local workstations. With massively increasing availability of EO data, the EO community is moving to an “exploitation platform” approach. Here, users and their applications are brought close to the physical location of EO data, helping to minimize data transfer between repositories and applications. To fully enable EO exploitation platforms, an interoperable, open standards-based architecture transformation is required.

Summary of Experiments

Initially, CRIM experienced difficulty to connect to the Amazon Web Services (AWS) instance through AWS SSM (Systems Manager Agent) instead of plain SSH as per NRCan’s security requirement. One needs to first connect via AWS-CLI to the AWS instance running the EODMS-flavored ADES/EMS. The server configuration can then be accessed. In contrast, NRCan’s PBC requires two-step SSH: one to access the Boreal Cloud proxy, and another to connect to the actual server instance from there. Other instances of EODMS are directly accessed via a single SSH to the server.

Per NRCAN’s requirement, AWS Elastic Load Balancer (ELB) was used instead of direct HTTPS connection to the services, even if when running single instances of each services. This caused difficulty with health check configuration. If ELB check fails, calls to EMS/ADES servers fail 50% of time. Health check is mandatory with ELB. Some components need to communicate between each other through the actual public URL, which normally goes through ELB. Due to NRCAN’s security policy, it was impossible to connect to ELB from the instances. A workaround had to be applied by using Docker’s *extra_hosts* options and disable SSL certificate validation for those internal calls.

Currently, only one server exists behind the ELB of each ADES/EMS. Had it been required or desired to have several parallel deployments, several issues would need to be considered. First, major refactoring of the server configuration would be required to support a separate database from the other services, so it can be shared across multiple load-balanced instances. Second, multiple tests would be required to ensure no incoherent operations occur, such as race conditions of read/write to the shared database. Other considerations such as common file storage and shared Docker images cache for coherent behavior across replicated instances would also be to be addressed

Generic *docker-compose* for EMS/ADES is in a private git repository but could be shared. There is minimal amount of sensitive info (if any), as most is dummy information to be replaced by actual deployment. The generic configuration is implemented using template files, that generate actual configurations loaded to run the server after substitution of specific server configurations. Specific server configurations are unique for each deployment (every combination of ADES/EMS and location: EODMS, PBC, CRIM, etc.). The specific configuration is "side-by-side" to the generic ones and overrides the needed parameters to generate the specific configuration files from the templates.

Docker images of the services (both specific *Weaver*, *Magpie*, *Twitcher*, as well as generic database, proxy, etc.) are all public, and used for installation and maintenance by pulling them from their respective location via *docker-compose* definition. Issues can be added to Weaver [https://github.com/crim-ca/weaver], Twitcher [https://github.com/Ouranosinc/twitcher], Magpie [https://github.com/Ouranosinc/Magpie] GitHub repositories. Once corrected, merged and tagged, the tagged version launches a

Docker build automatically. The generated image is pushed to the Docker repository for use by stakeholders. Specific server configurations need to be updated with desired versions of each service, according to desired functionalities.

Both TIE#1002 and TIE#1003 are Stacker app execution tests, and they both fail due to SNAP as in NRCan PBC's case. Additionally, TIE#1102 is for ML Segmentation app and TIE#1303 for basic JSON-to-NetCDF converter are planned but data was not provisioned in time to execute them. CRIM is standing by for more use cases from NRCan in order to deploy application and test their execution.

AWS S3 Buckets

In order to support EODMS Amazon infrastructure, Weaver was modified to take the S3 bucket location as the input/output location of processes. S3 Buckets were not trivial to connect to EODMS' Amazon Virtual Machines (VMs), in part due to the very high level of security. Once S3 Buckets were accessible to servers, connection configurations and credentials had to be mounted to Weaver in order to actually allow fetching/storing input/output files to S3 buckets. For process execution inputs, a simple S3 endpoint is required, as normally done for HTTP(S) file references. For outputs, Weaver needs to be configured with *weaver.wps_output_bucket* = <*s3-bucket*>. Aside from providing the bucket reference, other capabilities were developed, for instance parsing the reference, retrieving AWS configuration and staging files locally for use by the executed process. Sample configurations employed for CRIM and EODMS instances can be found in Annex H.

6.2.5. NRCan Intentions

With two Weaver platforms now deployed inside the NRCan's EO data value stream, one upstream with EODMS and LEVEL 0, 1 data and one downstream in the PBC with LEVEL 2 value-add data, NRCan will be looking to invest in two main avenues:

- On-boarding of EODMS client workflows as ADES containers to operate in close proximity to the EO data it would normally have to download. Critical clients include the NRCan's Emergency Geomatics Service who provide critical, near real-time information to Public Safety Canada and emergency responders during ice break-up and flood events, ECCC's Canadian Ice Service who provide timely and accurate information about ice in Canada's navigable waters and the Near Real Time Ship Detection services from the Department of National Defence (DND).
- Interoperability of the EODMS-WEAVER and PBC-WEAVER to reciprocate value-added services using the standards that Weaver has enriched each platform with.

6.3. Climate Platforms

General information about ClimateData.ca [https://ClimateData.ca] can be found in Section 8. More information about experiments conducted with ESGF [https://esgf.llnl.gov/] can be found in ESGF Compute Challenge engineering report [1].

6.4. Interoperability Experiments

All interoperability experiments either planned or conducted by CRIM in this pilot project can be found in Annex G. Note that the TIE numbers in this table are used throughout this report to refer to specific issues or findings. Integration tests then evaluate EMS/ADES server configurations using various AP deployment and execution combinations in order to evaluate functional operation of intended server-specific behavior. For example, *TIE-1004* is presented below.

- Each TIE number in the TIE Table is prefixed on the corresponding test functions defined in the code to ease their identification.
- Tests are separated into different files in order to segregate configurations of the respective server parameters.
- JSON payloads for Application Packages and/or requests are provided in tests/resources.

TIE #1044 python code

```
@pytest.mark.ADES
@pytest.mark.CRIM   # app & server
@pytest.mark.application
@pytest.mark.Deimos
@pytest.mark.ProbaV
@pytest.mark.RS2
@test_steps("deploy", "submit", "execute", "stage-out")
def TIE_1004_CRIM_stacker_on_CRIM_ADES():  # noqa: C0103,N802
    """Similar to TIE-1000, but with different data sources."""
    cookies = login(CRIM_ADES_MAGPIE, "CRIM_ADES_USERNAME", "CRIM_ADES_PASSWORD")
    proc_id = deploy_process(CRIM_ADES_WEAVER, "crim-stacker-deploy.yml", cookies
=cookies, visibility=True)
    yield "deploy"
    job_id = execute_process(CRIM_ADES_WEAVER, "crim-stacker-execute-no-sentinel.yml",
proc_id, cookies=cookies)
    yield "submit"
    monitor_job(CRIM_ADES_WEAVER, proc_id, job_id, cookies=cookies)
    yield "execute"
    fetch_result(CRIM_ADES_WEAVER, proc_id, job_id, cookies=cookies)
    yield "stage-out"
```

6.5. Applications and Processes

6.5.1. Data Interfaces

The following data interfaces were used in the system, in the pilot or in previous initiatives.

- HTTPS
- OGC API - Features
- WMS, WFS, WCS
- S3 bucket
- local filesystem
- OPEnDAP
- OpenSearch

- EOImage

Data Types

It is to be noted that processes that take *files* as input require definition of a *MIME type*. During the experiments, the most prevalent data types were:

- ZIP
- GeoTIFF
- GeoJSON
- NetCDF
- Zarr

6.5.2. Applications

Most of CRIM's applications adopt a fan-in design, where an array of files is reduced to a single output. CRIM public application packages repository can be found on GitHub [https://github.com/crim-ca/application-packages/]. The AP contains the CWL descriptors that refers to the appropriate *execution unit*, in this case Docker images. The images are built on demand, and stored in a private Docker registry. The table below shows the source code location and the resulting Docker images. Images published in the */ogc-public* can be accessed, pulled and executed by anonymous users. Access to all other images requires appropriate credentials by end users or the ADES.

Table 2. Docker Registries

Source repo	Docker registry	Access	Description
https://www.crim.ca/stash/projects/OGC/repos/ogc-dockers	docker-registry.crim.ca/ogc-public	Public	SNAP/ML applications developed since OGC-TB14 project
https://www.crim.ca/stash/projects/OGC/repos/ogc-dockers	docker-registry.crim.ca/ogc	Private	SNAP preprocessing apps + EMS/ADES-related items

6.5.3. XML and JSON Bindings

CRIM employs JSON bindings for body of both the *deploy* and *execute* requests, as described in WPS-T 2.0 with REST/JSON bindings. Translation from XML to JSON, and vice-versa, is a matter of changing a library. With respect to Python programming, use of JSON files are considered a best practice. JSON files are supported natively by Python standard library. Additionally, their structure is very similar. Participants would benefit from selecting and maintaining common JSON parsing libraries and services.

6.5.4. Quoting and Billing

In this pilot, CRIM did not seek to improve its implementation of billing and quoting [https://docs.opengeospatial.org/per/18-050r1.html#_quotation_api] presented in Testbed-14. As stated in the

ADES & EMS Results and Best Practices ER [http://docs.opengeospatial.org/per/18-050r1.html], the complexity of quoting a workflow execution is very high. As the number of steps increase, the deterministic behavior of a workflow rapidly decrease. For example, assuming adequate description by application developers, it might be possible to infer estimates of time required for a single application based on input data volume alone, for a specific type of machine. If the data output of that first application is meant to be an input for a second application, the subsequent estimates might vary wildly due to mounting uncertainty on data volume. Also, as the workflow gets larger, the parameter space grows, adding even more uncertainty.

6.6. Platform Architecture

6.6.1. EMS and ADES Responsibilities

Experiments with multiple sites indicate that EMS is a good practice, as it acts as a proxy to the ADES. As such, it provides more flexibility in security schemes. Technically, nothing would preclude from a service or user to call ADES API directly. By allowing registration of pre-deployed WPS and API routes, the EMS can also act as a federated process catalog. An EMS can take into account services, applications and processes provided by several ADES, facilitating its role of orchestrator of distributed workflows.

6.6.2. Security

- There was no use for WSO2 in this pilot, but the challenge of authentication in federated environments remains.
- The Policy Enforcement Point (PEP) of CRIM's solution is called *Twitcher*; as it acts a security proxy.
- Similarly, the EMS also acts as a proxy to the ADES, but not specifically or exclusively as security proxy.
- The roles of the PEP, the EMS process registry and the security proxy can overlap, but were not addressed specifically in this pilot.

6.6.3. Resource Access

Docker credentials

For Docker credentials and configs, see TIE-3000, where Pixalytics added CRIM as user in Docker repository. A file located at *~/.docker/config.json* is automatically picked up by Docker pull commands (standard file employed by Docker CLI). Weaver is therefore launched by mounting that file at the appropriate location after doing *docker login* command toward the private repository that contains targeted images in the Application Packages. Given proper credentials, this allows Weaver to run *docker pull* from the private repository since the user is authenticated within "~/.docker/config.json".

Request validation

A YAML file is loaded in Weaver to validate at run-time specific or invalid requests. A custom configuration specifies, for matched regex URLs, additional parameters to be provided or attributed to HTTP requests, such as custom headers, credentials, timeout duration, etc. File in question has an example is on GitHub [https://github.com/crim-ca/weaver/blob/master/config/request_options.yml.example].

Other

We also note the following:

- In order to connect to S3 buckets, CRIM adapted its ADES to conform to NRCan's infra IAM roles
- We note the TIE-4000 with Rhea Group, where data inputs where self-hosted, but without security certificates. Added config to trust the site, bypassing checks.
- It was challenging to manage, store, and inject credentials in apps, but even more in workflows
- No discussion for security in DAPA, or even in ADES-EMS flow. Security should be baked in an API.

6.6.4. Access Control Lists

- ACL is managed by the Magpie component, by users, groups, services and by resources: same approach as Testbed-14, but with additional fixes.
- Could restrict execution of a visible, deployed process by looking at quotas for example.
- By default, CRIM deploys an application as private (not visible). Visibility by a user is not the same as right to execute.
- Learning curve required to do authorized API requests
- Federated ACL are still useful in more advanced use cases. Review use of external Identity Providers, such as *Keycloak*. Could be connected through OpenID.
- Establish clear trust relationships between ADES an EMS.

6.6.5. Implementation

Open Source Software

See previous reports [2, 3, 1, 4, 5] for details on the platform implementations of *PAVICS*, *Birdhouse* and *Weaver* EMS-ADES, and on the machine learning and remote sensing applications. Also, readers should refer to the *Open Source software* subsections of Section 6, Section 7 and Section 8.

Configurations

Generic source configuration repository for server instances (OGC, NRCAN). OGC-based servers employ a generic configuration repository and are specialized with server-specific settings and/or overrides. Finch-based servers employ a similar *docker-compose* structure as other servers, but the generic configuration repository is different. Each server has the above 'generic' configuration under path ~/compose while the specific configurations are under *~/config*.

- Pull changes from *ems-ades-compose* under the ~/compose directory.
- Pull changes from branch master of the corresponding specific server configuration repository under the *~/config* directory.

- Run *server-compose up -d*. This step does not need a specific directory, as it finds its way from anywhere to the *docker-compose* in *~/compose*. Sometimes a *--force-recreate* parameter is needed to force the hand of docker-compose to regenerate some images. System link references from *~/compose* to relevant files in *~/config* are already defined to find required environment variables and overrides.

See here for more information on PAVICS configuration [https://pavics-sdi.readthedocs.io/en/latest/dev/index.html].

Chapter 7. Remote Sensing Application

This section presents experiments conducted with a general-purpose remote sensing application, specialized as an image stacker.

7.1. Overview

This application runs an acyclic graph of remote sensing operations. In this pilot, the graph describes a stacker that takes several images as input and produces a single co-registered multi-band output. Internally, the Graph Processing Tool (GPT) of the Sentinel Application Platform (SNAP) toolbox takes as input an XML document describing the remote processing graph. At runtime, the application automatically parses the CWL to map inputs and outputs onto an internal XML representation.

7.1.1. Purpose

Data from different missions, captured at different azimuth or time, need to be co-registered to a common reference before analysis. This registration takes into account the local topography of the terrain to compensate for visual aberrations and missing information from the sources. The purpose of a stacker is therefore to generate a set of analysis-ready data from heterogenous observational sources. The resulting file, a multi-band image, exhibits uniform spatial sampling projected on the same location. Akin to datacubes, the data can then be considered *analysis-ready*.

7.1.2. Data Types

The application's reader and writer operations are compatible with the file types presented in the table below.

Table 3. Data Types

Format	Description
BEAM-DIMAP	The standard BEAM I/O format. It comprises an XML header based on the SpotImage/CNES DIMAP schema and ENVI images for the raster data.
GeoTIFF	A widely used EO data format, e.g. for Quickbird, LANDSAT, SPOT.
NetCDF	A widely used EO data format. BEAM supports NetCDF files conforming to the NetCDF CF Metadata Convention.
HDF-EOS	BEAM supports the HDF-EOS profile (HDF4) used by NASA Ocean Color data products of SeaWiFS, MODIS, OCTS, CZCS, and the gridded MODIS L3 products.

7.1.3. Open Source Software

The main software packaged in the application is SNAP. Below are the main components used.

- Sentinel Toolbox [https://github.com/senbox-org] - SNAP, ESA's SentiNel Application Platform, version 7.x
- Sen2Cor [http://step.esa.int/main/third-party-plugins-2/sen2cor/sen2cor_v2-8/] - Processor for Sentinel-2 Level 2A product generation and formatting
- Snappy - Python interface from SNAP

For this application, CRIM also created a new toolbox, *CRIMTBX*, that can be easily added to the current SNAP modules. At the moment of writing, the source code of this toolbox is not yet released as open source software, but is available on demand. Below are the operations provided by the *CRIMTBX* toolbox and used by the application.

Table 4. Toolbox Operators

Operator Name	Short Description
StackCreationOp.java	Utility functions to create a stacked product from a series of input images
Collocate.java	Collocates two products based on their geocoding
MTAnalysisOp.java	Multi-Temporal Analysis operations
SfsOp.java	Structural Feature Set (SFS), including Standard Deviation, Mean, Maximum and Minimum
ThreshOp.java	Separates an image in two or more classes using Otsu's method

7.1.4. References

Previous versions of this application were introduced in Testbed-13 [6] and Testbed-14 [2]. The following references provide public information about components, techniques, algorithms and information relevant to this application.

- SNAP (Sentinel Application Platform) and the ESA Sentinel 3 Toolbox [7]
- Additional examples of classification and extraction of spatial features using SFS [8]
- Otsu's method to threshold SFS onto classes [9]

7.2. Inputs

The application uses a series of files pointing to imagery data products to be co-registered and stacked. All inputs defined in the CWL are first downloaded (or simply mounted) in the working directory by the CWL runner. As shown in the code excerpt below, each image input is then mapped in a temporary internal XML file descriptor used by SNAP's *Read* operation.

Excerpt of the inputs generated in the XML graph

```
{
<graph id="Graph">
  <version>1.0</version>
  <node id="Read">
    <operator>Read</operator>
    <sources/>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <file>./RS2_OK18072_PK188251_DK178156_F22_20090501_110525_HH_HV_SLC.zip</file>
    </parameters>
  </node>
  <node id="Read(2)">[...]</node>
  <node id="Read(3)">[...]</node>
  <node id="Read(4)">[...]</node>
  <node id="Read(5)">[...]</node>
  [...]
</graph>
}
}
```

The stacker application was tested using **Deimos**, **PROBA-V**, **Radarsat-2**, **Sentinel-1** and **Sentinel-2** data products found in a common ROI over Montreal. As the *RS2_OK18072_PK188251_DK178156_F22_20090501_110525_HH_HV_SLC.zip* product is defined first, it is used as a common referential. Below are the samples used:

Table 5. Remote Sensing Samples

Name	Description
RS2_OK18072_PK188251_DK178156_F22_20090501_110525_HH_HV_SLC.zip	RADARSAT2 product, HH HV SLC
Deimos.tif	Deimos image, unidentified data product
PROBAV_S1_TOA_X10Y02_20150617_1KM_V101.tif	PROBAV data product
S1A_IW_GRDH_1SDV_20170519T224357_20170519T224422_016657_01BA57_B4F7.SAFE.zip	Sentinel-1 data product, GRDH
S2A_MSIL1C_20191212T155641_N0208_R054_T18TXR_20191212T191005.zip	Sentinel-2 data product

7.3. Processing

All dependencies required to install SNAP are packaged in an image named *esa-snap-install-7*. In turn, this image serves as a base for *esa-snap-proc-7*, which offers helper code to interface with GPT. Finally, from this last image is derived *snap7-stack-creation*. This image contains the CWL descriptor file as well as the parser that produces the internal XML representation.

Below is the entry point of the *snap7-stack-creation* Docker file. In this file, *parse.py* first generates the temporary *graph.xml* before calling GPT, effectively starting execution of the application.

Entry Point Script

```
#!/usr/bin/env bash
set -ex

GRAPH="/tmp/graph.xml"
CUR_DIR=$(dirname $(realpath $0))

echo "Generating process graph"
${CONDA_ENV}/bin/python ${CUR_DIR}/parse.py --graph "${GRAPH}" "$@"
echo "Processing graph"
gpt ${GRAPH}
echo "Process complete"
```

In the graph, the stacking operation by itself is contained in the operator *StackCreation* of the *CRIMTBX* toolbox. The first data product is chosen as reference for projection and resampling operations on all other bands from all other products. Bands are projected and resampled on the same geographic coordinate system and spatial resolution of the reference band. The intersection between the different bands is determined, and the bands are then extracted according to this intersection. The code excerpt below shows the *sourceProduct* mapping of read data products as inputs for the *StackCreation* operation.

Excerpt of the processing operation generated in the XML graph

```
{
<graph id="Graph">
  <version>1.0</version>
  <node id="Read">[...]</node>
  #for all images found in CWL inputs, do a node read
  [...]
  <node id="StackCreation">
    <operator>StackCreation</operator>
    <sources>
      <sourceProduct refid="Read"/>
      #for all images read, map sources
      [...]
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement"/>
  </node>
  <node id="Write">[...]</node>
</graph>
}
```

7.3.1. Parameters

The application’s interface has been kept intentionally simple, with a minimal number of parameters. While this led to simpler packaging, it causes limitations when using multiresolution products as reference images. In this implementation, only the metadata of the first band is used as a reference. In case of products with multiple resolutions, for example Sentinel2 at 10-20-60 meters, it is not possible to use a 10m resolution because these are not the first band.

7.4. Outputs

Excerpt of the writing operation generated in the XML graph

```
{
<graph id="Graph">
  <version>1.0</version>
  <node id="Read">[...]</node>
  #for all images found in CWL inputs, do a node read
  [...]
  <node id="StackCreation">[...]</node>
  <node id="Write">
    <operator>Write</operator>
    <sources>
      <sourceProduct refid="StackCreation"/>
    </sources>
    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
      <file>./stacker_output.tif</file>
        <formatName>GeoTIFF</formatName>
    </parameters>
  </node>
</graph>
}
```

The output is a single file, by default named *stacker_output.tif*, containing separate bands for each input data source. The following figure shows an RGB visualization of the output bands. From left to right, **Radarsat-2**, **Sentinel-1**, **Sentinel-2**, **ProbaV**, **Deimos**.

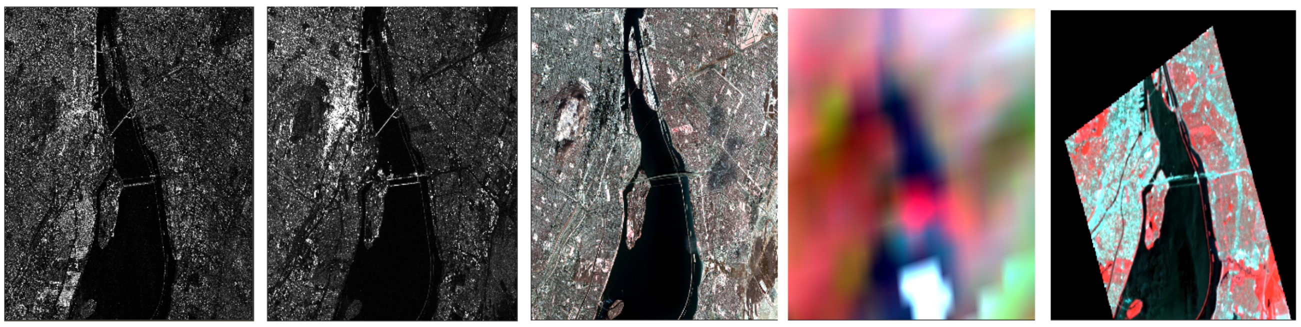


Figure 4. Image stack produced

Chapter 8. Machine Learning Application

This section presents a summary of experiments conducted with a trained model operating on satellite imagery.

8.1. Overview

This application uses a helper module called *thelper* to run model inference on an input raster image using a sliding window. A short annotation campaign was conducted before training of a model using *PyTorch*. The learned model classifies Sentinel-2 into 8 separate land use classes.

8.1.1. Purpose

The purpose of this application is to determine the most likely class for each pixel of a Sentinel-2 image. The figure below shows annotations that were used to train the model.

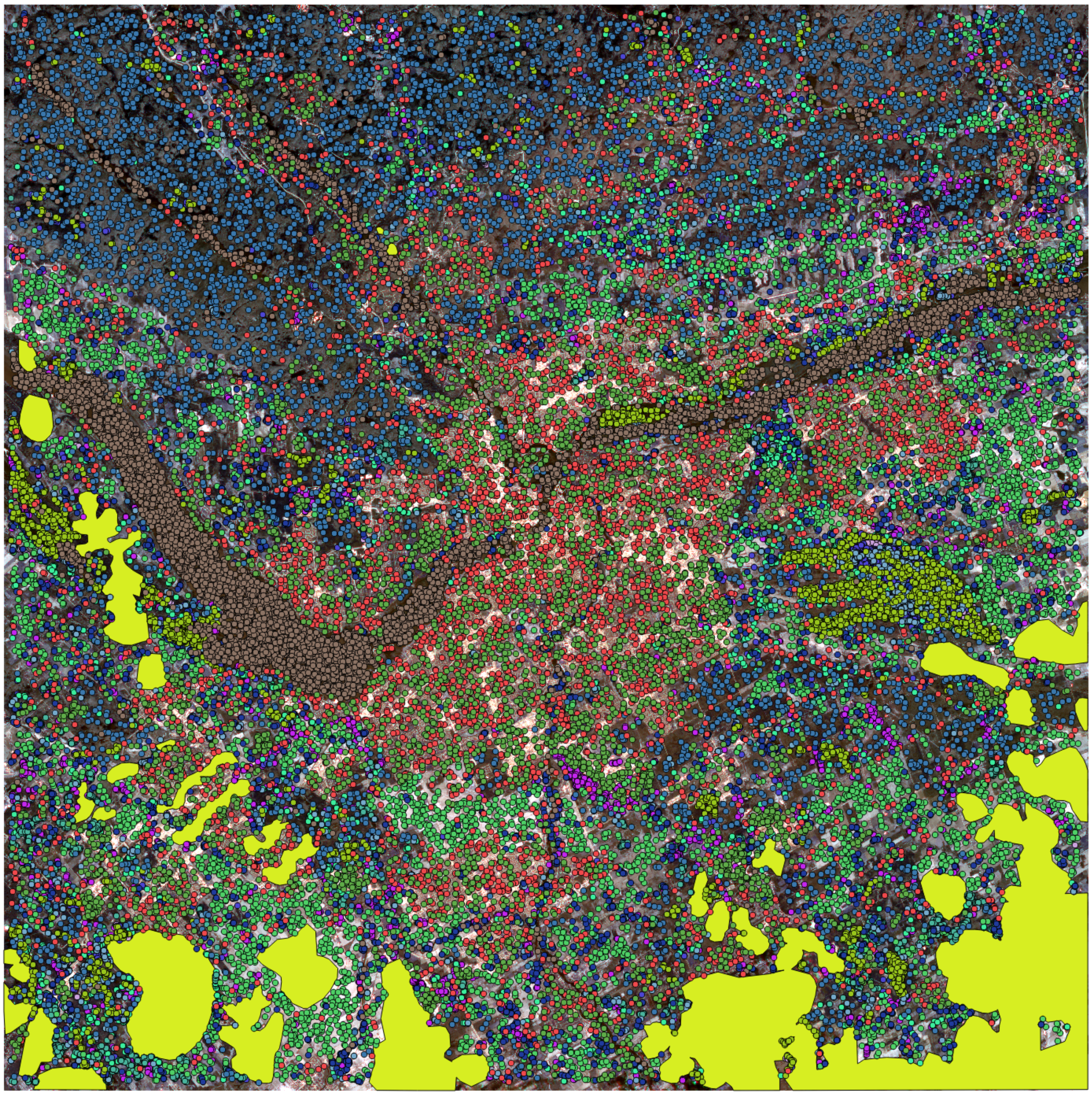


Figure 5. Annotation of land use

Once trained, the model output classes or *labels* for each pixel of an input image.

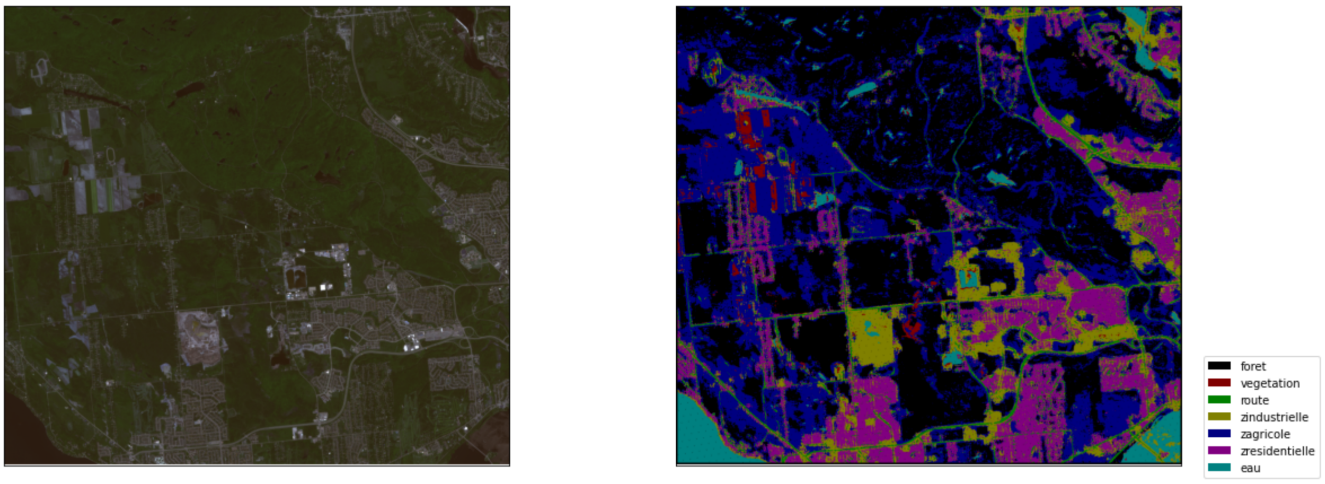


Figure 6. Model inputs and outputs

8.1.2. Data Types

As the model was trained with *Sentinel-2 L2A* data, the application only supports this data source. Using any other EO data would produce very noisy outputs.

8.1.3. Open Source Software

Below are the main components used in this application.

- PyTorch [https://pytorch.org/get-started/locally/] - An open source machine learning framework.
- thelper [https://github.com/plstcharles/thelper/] - A training framework and CLI for PyTorch-based machine learning projects.

The source code for the application package is not yet open source, but is available on demand. It can be found on CRIM private code repository [https://www.crim.ca/stash/projects/OGC/repos/ogc-dockers/browse/ogc-thelper-tb16]. Compiled Docker images from this code can be found in at CRIM private Docker registry [https://docker-registry.crim.ca/ogc/ogc-thelper-tb16].

8.1.4. References

Previous versions of this application were introduced in Testbed-14 [4] to demonstrate classification of Very High Resolution imagery. It was further developed in Testbed-15 [5] to train and execute a lake-river discrimination model.

Additionally, the following references provide public information about components, techniques, algorithms and background information relevant to this application.

- Land Use 1990, 2000 & 2010 [https://open.canada.ca/data/en/dataset/18e3ef1a-497c-40c6-8326-aac1a34a0dec]
- EuroSat dataset [10]
- Machine Learning, Deep Learning & Artificial Intelligence in the Open Geospatial Consortium Testbed-14
- Interoperable machine learning for Earth observation and climate in federated cyber infrastructures

8.2. Inputs

Sentinel-2 data packed as a ZIP archive was used as input for the inference process. The AOI considered is illustrated below.

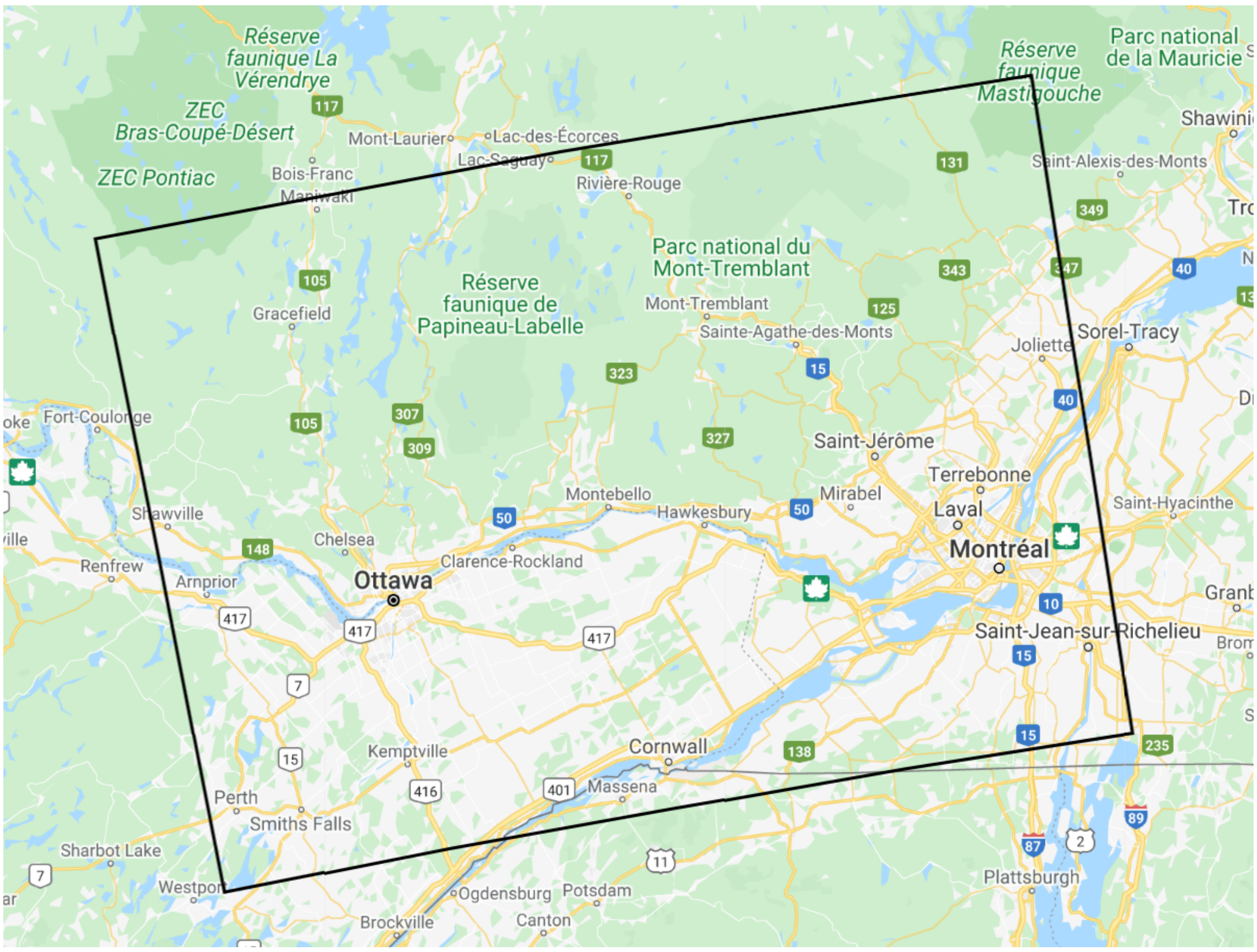


Figure 7. AOI for Machine Learning experiments (map © Google)

8.3. Processing

These results were obtained by running the *cwltool-run.sh* script and using a Sentinel-2 ZIP package as input. Process on Weaver ADES instance. Could require Magpie login for some URL.

- Deployed process: https://ogc-ades.crim.ca/ADES/processes/ogc-tb16-land-cover-map
- Application Package: https://ogc-ades.crim.ca/ADES/processes/ogc-tb16-land-cover-map/package
- Sample job execution: https://ogc-ades.crim.ca/ADES/processes/ogc-tb16-land-cover-map/jobs/b854b803-a32b-444f-b140-799be4a8365c

A sample job YAML is also provided to demonstrate a similar *cwltool* call to the scripts that would be called within the generated Docker image.

8.3.1. Running the Docker Image

The built Docker image can be called as follows:

```
docker run -ti ogc-thelper-tb16[:tag] [ogc_thelper_tb16 --help]
```

This will print the help usage message and argument descriptions by default. Additionally, *thelper* can be called directly using an explicit override of the executed command as follows:

```
docker run -ti ogc-thelper-tb16[:tag] thelper [--help]
```

This second command will result in exactly the same result as calling the *ogc-thelper-base* image or the original geo-enabled *thelper* Docker image.

8.3.2. Running the Application Package

Using cwltool, the Application Package can be tested with the following command:

```
cwltool process-application.cwl process-job.yml
```

This call is similar to how an ADES would actually call the process from a WPS-T REST call. As it is a manual test, the *process-job* file needs to be modified to provide the path to valid input files. Running the image with *cwltool* will pull the tagged remote Docker image as required, and then call it with the specified job arguments. If not already generated on the remote repository, *remote tag* can be created locally by calling the following command, which will build and tag the latest version for future push.

```
./build.sh tag
```

8.3.3. Parameters

In this case, the Docker image also contains the JSON body of process description and matching deployment CWL definitions in order to generate a compatible AP.

Raster File

The main parameter of the process is the *raster_file*. This corresponds to the raster reference that will be employed to run the model inference by running a sliding window over it. When running the process locally, either via direct Python call to *run.py* or literal Docker *run* command, the raster can be specified as either the direct XML product reference or the more generic ZIP package reference. In both situations, *GDAL* should handle the reference type automatically and retrieve the data accordingly.

```
docker run -ti \
        -v <path>/result:/workspace/result \
        -v <path>/S2A_[...]_DIR.SAFE:/workspace/raster \
        -ti ogc-thelper-tb16 thelper infer /workspace/raster/<product>.xml \
        -o /workspace/result \
        [<other-arguments>]
```

GPU Device

An optional parameter, *device=[cpu|gpu]*, allows you to specify where to run the code. If running with default values, the process will run model inference on CPU to make sure it works on any machine, but this could be very slow. If the host machine has access to a GPU (NVIDIA CUDA), the AP can make use of it to greatly increase processing speed. To employ the GPU within the Docker container, the driver needs to be configured in order to support CUDA toolkit support.

Batch and Patch Sizes

Another important consideration when defining the process execution is the values of parameters *batch_size* and *patch_size*. The *batch_size* will indicate how many samples to process during model inference at the same time. In this case, each sample is generated by a sliding window of *patch_size* x *patch_size* pixels. Therefore, the bigger the batch and patch sizes, the more pixels get covered simultaneously, but so does increase the memory consumption to load them. By default, *batch_size* = *1* to process samples one by one and patch_size is set according to model training.

Modifying the *patch_size* from the default value could greatly affect class inference accuracy as this effectively modifies the size of the image viewed by the model. For optimal results, it is often better to leave this dimension to the same value as it was employed during model training. If the host machine can load more data in memory, it is preferable to increase *batch_size* instead to process the result faster.

8.4. Outputs

The other GeoTIFF images and configuration files are the outputs of the CWL execution. The outputs of the application provide Land Cover Mapping of classes listed below, as well as recognition probabilities of every pixel within the presented input product. The model classifies the Sentinel-2 image into following classes:

- nodata: 0
- Forest: 1
- HerbaceousVegetation: 2
- Highway: 3
- Industrial: 4
- LandCrop: 5
- Residential: 6
- Water: 7

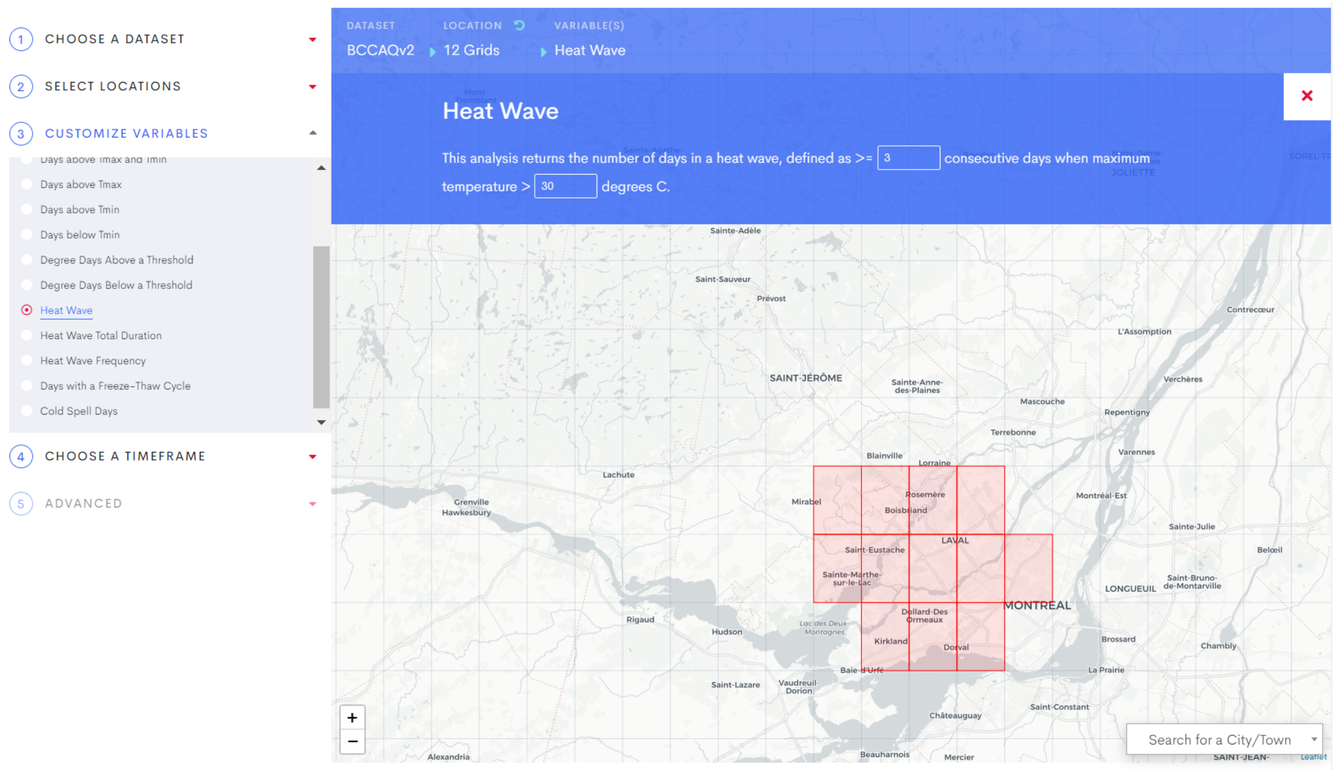
Chapter 9. Climate Services

This section presents an overview of experiments conducted with climate services, such as computation of climate indices from climate model outputs.

9.1. Overview

According to GFCS, *climate services* provide climate information to help individuals and organizations make climate related smart decisions. They equip decision makers in climate-sensitive sectors with better information to help society adapt to climate variability and change. Other definitions point to organizations that supply climate information to users.

In this pilot project, the climate services proposed by CRIM are intended to mature experiments previously done in the context of Testbed-14 for the benefit of the Earth System Grid Federation (ESGF [https://esgf.llnl.gov/index.html]). Experiments are also intended to advance analytic processes and APIs provided by the ClimateData.ca [https://www.ClimateData.ca] portal. ClimateData.ca is a climate information portal that enables Canadians to access, visualize, and analyze climate data, and provides related information and tools to support adaptation planning and decision-making. Below is a screen capture of the analysis page of the portal, where climate indices such as heat waves can be computed over fine-grained domains.


DATASET: BCCAQv2 · LOCATION: 12 Grids · VARIABLE(S): Heat Wave

Heat Wave

This analysis returns the number of days in a heat wave, defined as >= 3 consecutive days when maximum temperature > 30 degrees C.

Figure 8. Computation of heat waves for a region using ClimateData.ca

9.1.1. Purpose

Climate indices are not measured in the field or calculated by climate models. They are calculated or derived from climate variables such as temperature and precipitation, often through statistics or metrics related to individual grid cells. An exhaustive list of climate indices offered by *Finch* can be found in Annex B. Usually, indices can be more easily attributed to specific themes, sectors or domains than raw variables.

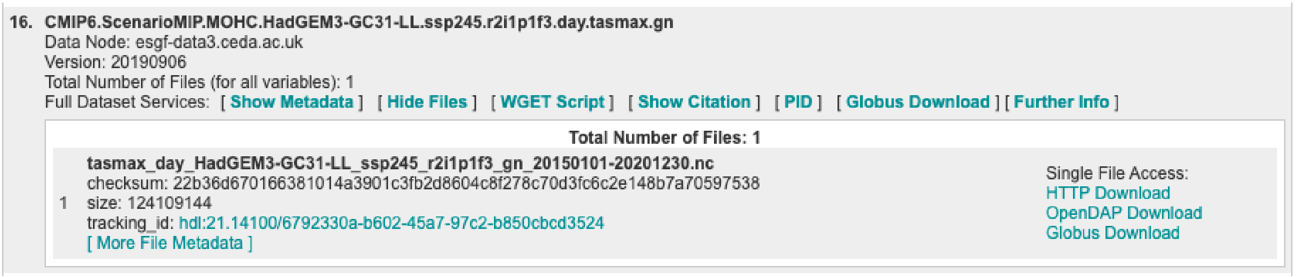
9.1.2. Data Types

The processes have been tested with the datasets presented in the table below. All climate data is stored in NetCDF files adhering to the CF Convention [https://cfconventions.org/index.html]. In all cases, the main variables used in the pilot are *temperature* and *precipitations*.

Table 6. Data Types

Name	Description
CMIP5	Coupled Model Intercomparison Project, Phase 5. Global, long-term climate model outputs. Usual spatial resolution of about 1.5°.
CMIP6	Coupled Model Intercomparison Project, Phase 6. Idem to CMIP5. High resolution models have spatial resolutions of 0.04166°, or about 5km square.
BCCAQv2	Statistically downscaled climate scenarios. Canada-wide, 10x10km long-term climate model outputs. Ensemble of 24 models considered. Spatial resolution of 0.08333°, or about 10km square.
ANUSPLIN	Gridded observational dataset. Canada-wide, 10x10km historical climate data modelled from station data. Spatial resolution of 0.08333°, or about 10km square.

The CMIP6 test data was acquired from ESGF CMIP6 Search [https://esgf-node.llnl.gov/search/cmip6/], using keywords *tasmax* and *scenariomip*. The main use case considered is the simple access and data reduction through a subsetting process. Below is a screen capture of the UI of the search feature. The data can be downloaded at tasmax_day_HadGEM3-GC31-LL_ssp245_r2i1p1f3_gn_20150101-20201230 [http://esgf-data3.ceda.ac.uk/thredds/fileServer/esg_cmip6/CMIP6/ScenarioMIP/MOHC/HadGEM3-GC31-LL/ssp245/r2i1p1f3/day/tasmax/gn/v20190906/tasmax_day_HadGEM3-GC31-LL_ssp245_r2i1p1f3_gn_20150101-20201230.nc].



16. CMIP6.ScenarioMIP.MOHC.HadGEM3-GC31-LL.ssp245.r2i1p1f3.day.tasmax.gn
Data Node: esgf-data3.ceda.ac.uk
Version: 20190906
Total Number of Files (for all variables): 1
Full Dataset Services: [Show Metadata] [Hide Files] [WGET Script] [Show Citation] [PID] [Globus Download] [Further Info]

Total Number of Files: 1

1 **tasmax_day_HadGEM3-GC31-LL_ssp245_r2i1p1f3_gn_20150101-20201230.nc**
checksum: 22b36d670166381014a3901c3fb2d8604c8f278c70d3fc6c2e148b7a70597538
size: 124109144
tracking_id: hdl:21.14100/6792330a-b602-45a7-97c2-b850cbcd3524
[More File Metadata]

Single File Access:
HTTP Download
OpenDAP Download
Globus Download

Figure 9. Data search from ESGF

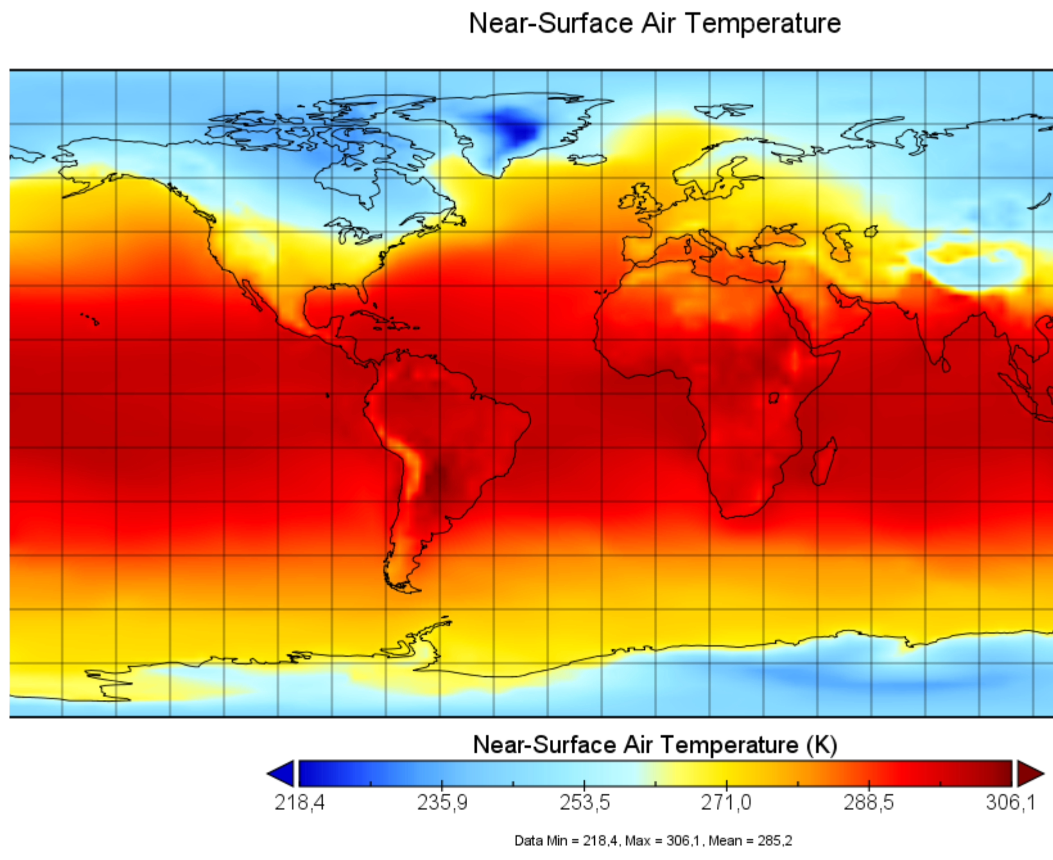

Near-Surface Air Temperature
Near-Surface Air Temperature (K)
218,4
235,9
253,5
271,0
288,5
306,1
Data Min = 218,4, Max = 306,1, Mean = 285,2

Figure 10. Daily max temperature CMIP6 sample selected

9.1.3. Open Source Software

Climate indices are computed from climate model outputs using the specialized *xclim* Python library. Xclim is deployed as a WPS endpoint using *Finch*, part of the *Birdhouse* framework. Additional base climate services and processes are also found in *PAVICS*, a climate research platform built on top of *Birdhouse*.

- FINCH [https://github.com/bird-house/finch] - A Web Processing Service for Climate Indicators, Ouranos/CRIM.
- XCLIM [https://github.com/ouranosinc/xclim] - A library of functions to compute climate indices from observations or model simulations, Ouranos.
- Birdhouse [http://bird-house.github.io/] Framework, DKRZ/CEDA/IPSL.
- PAVICS [https://ouranosinc.github.io/pavics-sdi/] - Platform for the Analysis and Visualization of Climate Science, Ouranos/CRIM.

As with any other "birds" of the *Birdhouse* climate framework, *Finch* is meant to be deployed as an autonomous WPS 1.0 endpoint served by *PyWPS*. An EMS implemented with *Weaver* can then register the external endpoint. Once registered, climate services can be discovered and executed using *OGC API - Processes*, or chained into workflows. The services are considered as *applications* in the sense that they are provided with appropriate descriptors allowing use in workflows and remote execution. Nevertheless, there is no deployment per se, the closest operation being registration in an EMS.

- WEAVER [https://github.com/crim-ca/weaver] - EMS/ADES implementation following OGC API - Processes best practices.

- PYWPS [https://github.com/geopython/pywps] - Implementation of the WPS standard.

PAVICS users tend to access to climate data through notebooks. The *Birdy* native Python client is built on *OWSLIB*. It allows the dynamic generation of a module whose functions look and feel like native python objects but actually execute remote processes. Scientists and developers can thus blend online WPS functionalities in ordinary scripts, offloading large computation tasks to external dedicated computing resources.

- OWSLIB [https://github.com/geopython/owslib] - Python package for client programming with OWS.
- BIRDY [https://github.com/bird-house/birdy] - Allows interaction with OWS in Python environments.

9.1.4. References

Previous versions of this application were introduced in Testbed-14 activities supported by U.S. Department of Energy's (DOE) Office of Biological and Environmental Research (BER) [1]. The latest architectural updates can be found in ESGF Future Architecture report [https://zenodo.org/record/3928223#.Xw8TpyhKi8N]. The report notes the following activities as relevant to ESGF.

NOTE

ESA has funded a number of initiatives relevant to ESGF. The HMA (Heterogeneous Missions Accessibility [https://earth.esa.int/hma/]) project seeks to harmonize and standardize ground segment services and interfaces. FedEO [http://wiki.services.eoportal.org/tiki-index.php?page=FEDEO] builds on the work of HMA and provides a brokered access to a distributed set of catalogues for EO data. A series of projects has sought to exploit cloud computing to address the challenges of Big Data - the SSEP (SuperSites Exploitation Platform) [http://wiki.services.eoportal.org/tiki-index.php?page=SSEP], TEPs (Thematic Exploitation Platforms) [https://eo4society.esa.int/thematic-exploitation-platforms-overview/] and EOEPCA (EO Exploitation Platform Common Architecture) [https://eoepca.github.io/]. The latter is currently underway and seeks to develop a common reference architecture for federating Earth observation platforms.

The following references provide public information about components, techniques, algorithms and contextual information relevant to this application.

- PAVICS [https://ouranosinc.github.io/pavics-sdi/]
- Birdhouse [http://bird-house.github.io/]
- BIRDY [https://github.com/bird-house/birdy]
- BCCAQv2 [https://www.climatedata.ca/about]
- ANUSPLIN [https://www.climatedata.ca/about]

9.2. Inputs

A complete JSON body for the execute request of the *subset_bbox* process can be found in Annex F. The actual process is described in the next section. The main inputs to the process are *resource* - the URI to a CMIP6 NetCDF-CF file and *lon0*,*lat0*,*lon1*,*lat1* - the two opposing corners of a rectangular bounding box.

The complete WPS 1.0 process description for *frost days* can be accessed through this link [https://pavics.climatedata.ca/wps?service=WPS&request=describeprocess&version=1.0.0&identifier=ice_days]. Below is the JSON body of a request sufficient to execute the computation.

JSON request body to execute frost days

```
{
  "mode": "async",
  "response": "document",
  "inputs": [
    {
      "id": "tasmax",
      "href":
"https://github.com/Ouranosinc/xclim/raw/master/tests/testdata/NRCANdaily/nrcan_canada
_daily_tasmax_1990.nc"
    },
    {
      "id": "freq",
      "data": "MS"
    }
  ],
  "outputs": [
    {
      "id": "output_netcdf",
      "transmissionMode": "reference"
    }
  ]
}
```

9.2.1. CMIP6 OpenSearch Endpoint

One technical objective of the climate services experiments is evaluation of *OpenSearch* as data acquisition and preparation step of a workflow. This use case is similar to Testbed-14 OpenSearch results [https://docs.opengeospatial.org/per/18-050r1.html#_opensearch_results_pagination].

This service can be found here [http://opensearch-test.ceda.ac.uk/opensearch/description.xml], while the code is on GitHub [https://github.com/cedadev/archive-opensearch]. The description document lists the search facets that it supports. This OpenSearch repository expose ESA Climate Change Initiative datasets. OpenSearch uses a drill down approach, where the search is done at the top-level with collections (datasets), then is followed down to granules (individual files). Below are a few sample requests:

- Description document for the cci project [http://opensearch-test.ceda.ac.uk/opensearch/description.xml?parentIdentifier=cci]
- Top-level collections [http://opensearch-test.ceda.ac.uk/opensearch/request?parentIdentifier=cci&httpAccept=application/geo%2Bjson]
- Description for a specific collection [http://opensearch-test.ceda.ac.uk/opensearch/description.xml?parentIdentifier=ce524139de81430e840d9a33daab3385]

- Search the last collection of files based on processing level [http://opensearch-test.ceda.ac.uk/opensearch/request?parentIdentifier=ce524139de81430e840d9a33daab3385&httpAccept=application/geo%2Bjson&processingLevel=L2]

9.3. Processing

In order to prepare the data for processing, the following basic subsetting processes from project *PAVICS* are available. In this project, only *subset_bbox* was used in workflows.

Table 7. Basic climate processes

Index	**Title**	**Description**
subset_grid_point_dataset	Subset of grid cells from a dataset, using a list of coordinates	For the given dataset, return the closest grid cell for each provided coordinates pair, for the time range selected.
subset_bbox_dataset	Subset of a dataset, using a bounding box	For the given dataset, return the data for which grid cells intersect the bounding box for each input dataset as well as the time range selected.
subset_bbox	Subset with bounding box	Return the data for which grid cells intersect the bounding box for each input dataset as well as the time range selected.
subset_gridpoint	Subset with a grid point	Return the data for which grid cells includes the point coordinates for each input dataset as well as the time range selected.
subset_polygon	Subset with one or more polygons	Return the data for which grid cells centers are within the polygon for each input dataset as well as the time range selected.

9.4. Outputs

Below is a visualization of the output generated by the *subset_bbox* process. In cases where an analytical application cannot be fully deployed near the data, this step provides a notable data reduction for any subsequent steps in a workflow.

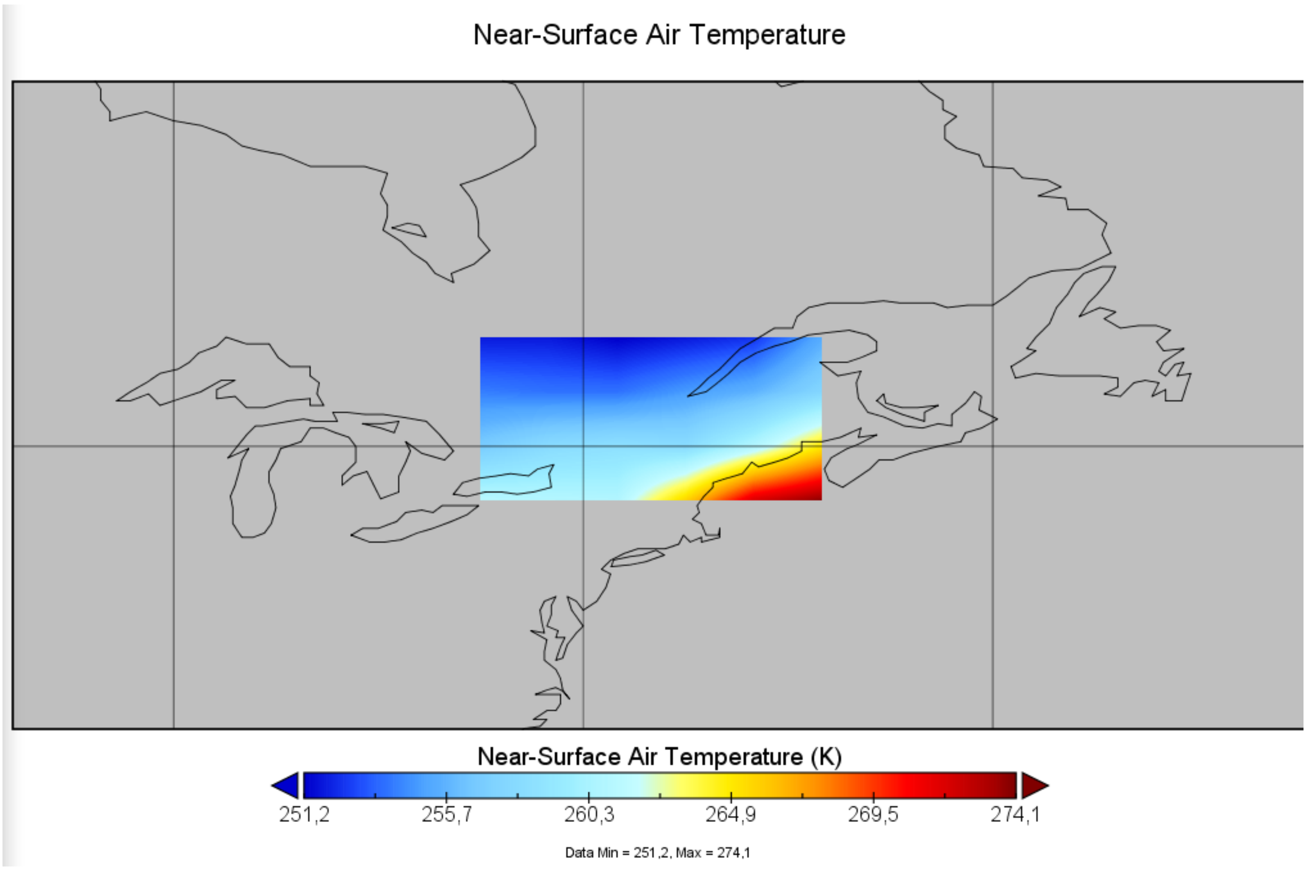
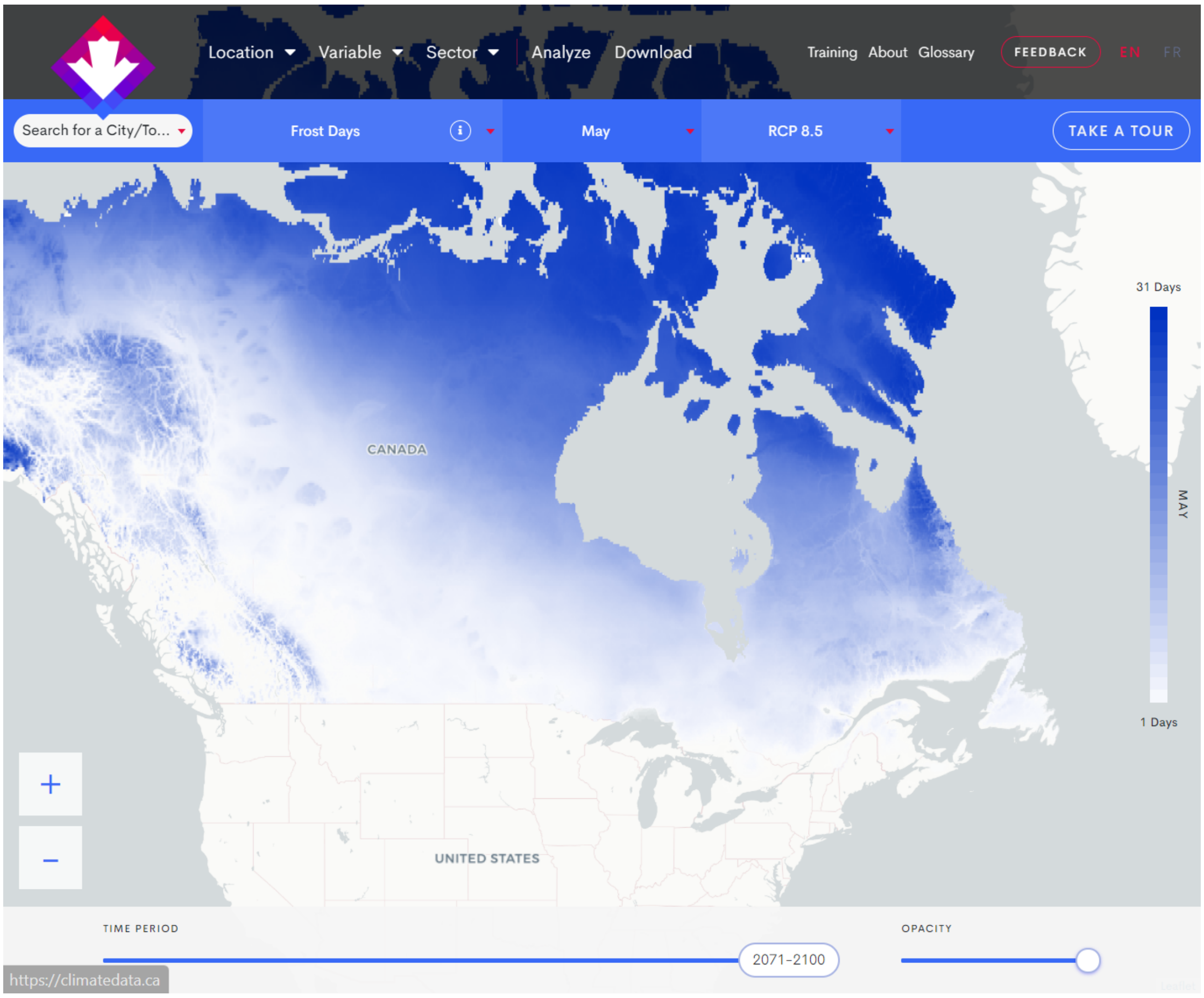

Near-Surface Air Temperature
Near-Surface Air Temperature (K)
251,2
255,7
260,3
264,9
269,5
274,1
Data Min = 251,2, Max = 274,1

Figure 11. Subset output from CMIP6 sample data

Below is a visualization of a typical output generated by the *frost days* climate index computation. This provides a quick assessment of the number of days where daily minimum temperatures are below 0° Celsius.



Frost Days

May

RCP 8.5

TIME PERIOD: 2071-2100

OPACITY

Figure 12. Frost days indicator output from BCCAQv2 data

Chapter 10. Earth Observation Applications

This section summarizes the main findings from an application developer perspective. It is meant as a generalization of the three applications presented in this pilot.

10.1. Application Definition

10.1.1. Application Template

CRIM notes the following practices pertaining to the template of an AP:

- Composition of an AP: CWL file and JSON body for both *Deploy* and *Execute*
- *ProcessDescription* (example here) [https://pavics-weaver.readthedocs.io/en/latest/processes.html#wps-rest] is generated partly from CWL (example here) [https://pavics-weaver.readthedocs.io/en/latest/package.html]
- The application is structured largely by the JSON *Deploy* body
- In the *Deploy* body, bare minimum parameters are *ProcessID* and *Execution unit*
- *Execution Unit* is defined in the CWL body, either as reference or in-lined
- The CWL refer to compiled Docker image, or other execution units such as script or pre-deployed WPS

10.1.2. WPS Interface Definition

The following remarks apply to the WPS interface of deployed applications:

- As soon as the application is deployed on ADES, a WPS definition is available
- This definition is accessible both in WPS-T 2.0 REST, and WPS 1.0
- An external WPS can be registered by the EMS, becoming an application-process registry
- Akin to ADES deployed apps, the EMS offers definitions in WPS-T 2.0 and WPS 1.0

10.1.3. Dependencies Management

For remote sensing apps, most dependencies (operations, libraries) are in Java as SNAP is the main component. Most problems were related to packaging of SNAP since there is not official ESA Docker image. Each participant has his own flavor which leads to inconsistent functionalities and expectations. Also, because each developer is left on his own to package SNAP, there have always been problems related to file/dir permissions, as most developers testing locally have close to root access while server CWL execution does not.

For machine learning apps, the main dependency is PyTorch. A Python helper library called *thelper* [https://github.com/plstcharles/thelper] is exposing the application's entry point. It allows the definition of training/inference of a given model over specified datasets, but providing metadata, chain of transformations and other properties via a YAML configuration. It helps repeatability and customization of the operations and tests without modifications to the code. Without this kind of parametrization, each researcher is left to himself to develop his custom ML workflow and scripts, which leads to many assumptions left undocumented or implementation choices hard to comprehend by following researchers/developers. Fully-parametrized applications helps guide researchers toward a somewhat common development mentality, which provides easier collaboration or extension opportunities over preexisting algorithms. Additional dependencies are pulled from *pip* and *Conda* at build time of the Docker image.

For climate services, *Birdhouse* framework provides individual WPS servers called *birds*. Each *bird* runs *PyWPS* and contains several related processes, such as computation of climate indices. In the framework, dependencies are also managed by combination of *pip* and *Conda*. The reuse of existing WPS processes accelerated development of functional application (compared to starting from scratch), but also highlighted the need to support older generation WPS. Although *OGC API - Processes* offer new functionalities and formats, sometimes these preexisting processes still need to be employed for a longer period. This led to the creation of WPS-1 and WPS-2 "converters" to new WPS-REST bindings.

The reader can also refer to external documentation found at PAVICS WPS 1 and 2 [https://pavics-weaver.readthedocs.io/en/latest/processes.html#wps-1-2] and PAVICS ESGF access [https://pavics-weaver.readthedocs.io/en/latest/processes.html#esgf-cwt]. These process definitions allow a common ADES interface, but offered for different kinds of applications such as AP or remote pre-existing, pre-deployed WPS. This also forced consideration of ADES and AP implementations, deployment mechanism and features offered as not exclusively and completely manageable as local Docker image.

10.1.4. Scripts and Executables

In a first scenario, the developed application is managed by the same institution that offer the application package. In that case, it is good practice for the developer to only provide a command-line interface to his executable. The developer is in charge of packaging then create a Docker image where the *run* command is mapped to the command-line interface. As described previously, the CWL then defines the *Execution Unit* to the Docker image. An advantage of this practice is that the application can be tested locally easily, without having to redeploy each time. The local execution is also similar to the distant execution on an ADES.

To test basic cases of ADES-EMS deployments, CRIM used an EO application calling a local script, instead of pulling a Docker image. This test app runs a script that extracts and fetches NetCDF files from a JSON file containing an array of URLs, and provides them on the output directory.

10.2. Discussion

10.2.1. Execution on an Exploitation Platform

In the case of applications, a Docker image is available in a registry. At *Deploy* time, the process is created on ADES and rendered visible. At *Execute* time, the image is pulled by the ADES and run, spawning a Docker container on a virtual machine.

For workflows, each step is registered as a process in the EMS. For each step, a *Deploy* request is sent to the appropriate ADES. Currently, and as it was the case for Testbed-14, the EMS explicitly links different mission data to specific ADES. We recommend that the EMS API provides a way to specify the target ADES. Ultimately, the EMS role could be extended to offer orchestration and optimization rules against a federated data and processing infrastructure.

10.2.2. Hardware Requirements

Applications could benefit from a formal description of their hardware requirements, for example GPU, multicore, memory. Inversely, the ADES would benefit from describing its capabilities and offerings. This way, at *Deploy* time, application requirements can be checked against ADES capabilities. Amongst advanced platform capabilities to potentially expose to the application user, we note the distributed execution environment, for example *Kubernetes* or *Amazon Elastic BeanStalk*.

10.2.3. Common Workflow Language

CRIM favors use of CWL in the Common Architecture for the following reasons:

Separation of Concerns

CWL provides a much clearer definition of which application gets deployed and how it gets executed. It is because of the explicit positioning of arguments/options and typing applied to input/outputs. This explicit definition removes the need for mixing in additional WPS-related functionalities that are not interacting directly with the *docker-run* call in the first place. CWL offers a clean separation between the WPS portion and the core software/hardware requisites related to "running the app" portion. The former should focus more on API schema definitions, supporting operations like deployment, cataloging, monitoring and file transfer.

Documentation and Community

CWL's documentation and core specification are extremely advanced and complete. The community is large and functionalities are greatly evaluated on a regular basis, making the format more reliable and full-fledged according to real use-cases. Other approaches such as employing *additionalParameters* are not as verbose and do not provide any reference specifications nor examples to help newcomers using this approach. On the other hand, CWL documentation offers all these and most problematic use cases are already covered by a solution in examples due to previously encountered experiences by their users.

Development Cycle

Finally, CWL allows the AP developer to extensively and rapidly test his app locally until satisfying results are obtained without constant re-deploy/execute/undeploy for every tweak. This is because AP deployed via CWL are executed in the same manner (*cwltool*) both locally and server-side under-the-hood by ADES/EMS. This greatly minimizes time and complexity involved during conception and testing phases of the app. Also, it requires a lower learning curve from the developer as he/she can only focus on his app, instead of the full WPS+CWL chain.

Chapter 11. Conclusion

This section concludes the ER by presenting general findings and recommendations pertaining to EO platforms and applications.

11.1. EO Platforms

A general objective of CRIM's proposal in this pilot was to evaluate both EO and climate cyber infrastructures, and to have a better understanding of their respective gaps with respect to the common architecture. It is safe to say that this objective is partially met from the platform manager perspective.

The core ADES-EMS implementations were deployed in three different sites, *PBC*, *EODMS* and CRIM. The interoperability testing was also advanced with climate data providers, such as ClimateData.ca, CEDA, Ouranos and DKRZ. The use of an application profile to register external WPS proves to be a rather simple, but useful tool to build a federated cloud.

The separation of roles between EMS and ADES is still seen as beneficial, the former being closer to thematic platforms with cross-cutting concerns, the latter located near mission data, or more specifically to the ground segment. We also note that producing accurate quotations for workflows will be a very difficult challenge, while quoting for applications in simple federations is probably feasible.

11.1.1. Recommendations

- Evaluate the potential Machine Learning services (training, annotation management, etc.) that platforms should offer to stimulate and support creation of ML applications, and integrate findings into EOEPCA.
- Develop and adopt API specifications for distributed processing back-end offerings, for example Kubernetes, SLURM or GPU-enabled VM.
- Continue implementation of data connectors (S3 buckets, Zarr, etc.) and expose these platform capabilities into the conformance route of ADES instances.
- Increase test coverage for configurations, applications, platforms and federations.
- Continue maturation of the common architecture through new federated cloud projects, with intent of advancing distributed multi-disciplinary workflows. By extension, continue development of best practices for hybrid authorization and authentication schemes.

11.2. EO Application

Another general objective of CRIM's proposal in this pilot was to evaluate both EO and climate real-world applications, and to assess their fitness with respect to the common architecture. It is safe to say that this objective is met from the application developer perspective.

A positive outcome for the project was the reassessment of the ease of use and fitness of CWL for the common architecture. The remaining issues and gaps between participants are mostly on data and query interfaces (XML, JSON, POST vs GET), and on what CWL attribute should be supported.

Generally, we also note that the Docker ecosystem is still robust and appropriate for development of application marketplaces. There is an increasing availability of distributed execution framework that could support operationalization and integration to DevOps.

11.2.1. Recommendations

- Consider CWL as a potential core standard for OGC API - Processes to define geospatial application and workflows.
- Develop and adopt API specifications for hardware requirements of applications, in conjunction with EO platform capabilities presented in previous sub-section.
- Strive to develop common core, stable base Docker images, that could be used to rapidly derive specific applications.
- Evaluate the fitness of the prototype Data Access and Processing API (DAPA) and the OGC API - Environmental Data Retrieval [https://www.ogc.org/standards/requests/215] candidate standard in the development of AP.

Appendix A: Climate Indices Available Through Finch

Table 8. Climate indices

Indice	Title	Description
rain_frzgr	Number of rain on frozen ground days	Number of days with rain above a threshold after a series of seven days below freezing temperature.
rx1day	Maximum 1-day total precipitation	Resample the original daily total precipitation temperature series by taking the max over each period.
max_n_day_precipitation_amount	Maximum {window}-day total precipitation	Calculate the n-day rolling sum of the original daily total precipitation series and determine the maximum value over each period.
wetdays	Number of wet days (precip >= {thresh})	Return the total number of days during period with precipitation over threshold.
dry_days	Number of dry days (precip < {thresh})	The number of days with daily precipitation below threshold.
cdd	Maximum consecutive dry days (Precip < {thresh})	Return the maximum number of consecutive days within the period where precipitation is below a certain threshold.
cwd	Maximum consecutive wet days (Precip >= {thresh})	Returns the maximum number of consecutive wet days.
sdii	Average precipitation during Wet Days (SDII)	Return the average precipitation over wet days.
prcptot	Total precipitation	Resample the original daily mean precipitation flux and accumulate over each period.
liquidprcptot	Total liquid precipitation	Resample the original daily mean liquid precipitation flux and accumulate over each period.

Indice	Title	Description
solidprcptot	Total solid precipitation	Resample the original daily mean solid precipitation flux and accumulate over each period.
tn_days_below	Number of days with Tmin < {thresh}	Number of days where daily minimum temperature is below a threshold.
tx_days_above	Number of days with Tmax > {thresh}	Number of days where daily maximum temperature exceed a threshold.
tx_tn_days_above	Number of days with Tmax > {thresh_tasmax} and Tmin > {thresh_tasmin}	The number of days per period with tasmin above a threshold and tasmax above another threshold.
heat_wave_frequency	Number of heat wave events	Number of heat waves over a given period.
heat_wave_max_length	Maximum length of heat wave events	-
heat_wave_total_length	Total length of heat wave events	Total length of heat waves over a given period.
heat_wave_index	Number of days that are part of a heatwave	Number of days that are part of a heatwave, defined as five or more consecutive days over 25.
hot_spell_frequency	Number of hot spell events	Number of hot spells over a given period.
hot_spell_max_length	Maximum length of hot spell events	-
tg Daily	mean temperature	We assume a symmetrical distribution for the temperature and retrieve the average value as Tg = (Tx + Tn) / 2
tg_mean	Mean daily mean temperature	Resample the original daily mean temperature series by taking the mean over each period.
tg10p	Number of days when Tmean < 10th percentile	Number of days with daily mean temperature below the 10th percentile.

Indice	Title	Description
tg90p	Number of days when Tmean > 90th percentile	Number of days with daily mean temperature over the 90th percentile.
tn_min	Minimum daily minimum temperature	Minimum of daily minimum temperature.
tn_max	Maximum daily minimum temperature	The maximum of daily minimum temperature.
tn_mean	Mean daily minimum temperature	Mean of daily minimum temperature.
tn10p	Number of days when Tmin < 10th percentile	Number of days with daily minimum temperature below the 10th percentile.
tn90p	Number of days when Tmin > 90th percentile	Number of days with daily minimum temperature over the 90th percentile.
tx_min	Minimum daily maximum temperature	The minimum of daily maximum temperature.
tx_max	Maximum daily maximum temperature	The maximum value of daily maximum temperature.
tx_mean	Mean daily maximum temperature	The mean of daily maximum temperature.
tx10p	Number of days when Tmax < 10th percentile	Number of days with daily maximum temperature below the 10th percentile.
tx90p	Number of days when Tmax > 90th percentile	Number of days with daily maximum temperature over the 90th percentile.
dtr	Mean Diurnal Temperature Range	The mean difference between the daily maximum temperature and the daily minimum temperature.
dtrvar	Mean Diurnal Temperature Range	Variability Mean absolute day-to-day variation in daily temperature range.
etr	Intra-period Extreme Temperature Range	The maximum of max temperature (TXx) minus the minimum of min temperature (TNn) for the given time period.

Indice	Title	Description
cold_spell_duration_index	Cold Spell Duration Index	Number of days with at least six consecutive days where the daily minimum temperature is below the 10th percentile.
cold_spell_days	cold spell index	The number of days that are part of a cold spell, defined as five or more consecutive days with mean daily temperature below a threshold in degC.
dlyfrzthw	daily freezethaw cycles	The number of days where Tmax > thresh_tasmax and Tmin ⇐ thresh_tasmin.
cooling_degree_days	Cooling Degree Days (Tmean > {thresh})	Sum of degree days above the temperature threshold at which spaces are cooled.
growing_degree_days	growing degree days above {thresh}	The sum of degree-days over the threshold temperature.
heating_degree_days	Heating Degree Days (Tmean < {thresh})	Sum of degree days below the temperature threshold at which spaces are heated.
freshet_start	Day of year of spring freshet start	Returns first day of period where a temperature threshold is exceeded over a given number of days.
frost_days	Number of Frost Days (Tmin < 0C)	Number of days where daily minimum temperatures are below 0.
ice_days	Number of Ice Days (Tmax < 0)	Number of days where daily maximum temperatures are below 0.
consecutive_frost_days	Maximum number of consecutive days with Tmin < 0C	Resample the daily minimum temperature series by computing the maximum number of days below the freezing point over each period.

Indice	Title	Description
consecutive_frost_free_days	Maximum number of consecutive days with Tmin > {thresh}	Return the maximum number of consecutive days within the period where the minimum temperature is above a certain threshold.
tropical_nights	Number of Tropical Nights (Tmin > {thresh})	The number of days with minimum daily temperature above threshold.

Appendix B: Process Description of Finch

Process description

```
{
<wps:ProcessDescriptions
    xmlns:wps="http://www.opengis.net/wps/1.0.0"
    xmlns:ows="http://www.opengis.net/ows/1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
"http://www.opengis.net/wps/1.0.0 ../wpsDescribeProcess_response.xsd" service="WPS"
version="1.0.0" xml:lang="en-US">
    <ProcessDescription wps:processVersion="0.1" storeSupported="true"
statusSupported="true">
        <ows:Identifier>ensemble_grid_point_dlyfrzthw</ows:Identifier>
        <ows:Title>daily freezethaw cycles</ows:Title>
        <ows:Abstract>
The number of days where Tmax > thresh_tasmax and Tmin <= thresh_tasmin.

        </ows:Abstract>
        <DataInputs>
            <Input minOccurs="1" maxOccurs="1">
                <ows:Identifier>lat</ows:Identifier>
                <ows:Title>Latitude</ows:Title>
                <ows:Abstract>
Latitude coordinate. Accepts a comma separated list of floats for multiple grid cells.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                </LiteralData>
            </Input>
            <Input minOccurs="1" maxOccurs="1">
                <ows:Identifier>lon</ows:Identifier>
                <ows:Title>Longitude</ows:Title>
                <ows:Abstract>
Longitude coordinate. Accepts a comma separated list of floats for multiple grid
cells.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>start_date</ows:Identifier>
                <ows:Title>Initial date</ows:Title>
                <ows:Abstract>
Initial date for temporal subsetting. Can be expressed as year (%Y), year-month (%Y-
%m) or year-month-day(%Y-%m-%d). Defaults to first day in file.
```

```
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>end_date</ows:Identifier>
                <ows:Title>Final date</ows:Title>
                <ows:Abstract>
Final date for temporal subsetting. Can be expressed as year (%Y), year-month (%Y-%m)
or year-month-day(%Y-%m-%d). Defaults to last day in file.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>ensemble_percentiles</ows:Identifier>
                <ows:Title>Ensemble percentiles</ows:Title>
                <ows:Abstract>
Ensemble percentiles to calculate for input climate simulations. Accepts a comma
separated list of integers.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <DefaultValue>10,50,90</DefaultValue>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>dataset_name</ows:Identifier>
                <ows:Title>Dataset name</ows:Title>
                <ows:Abstract>
Name of the dataset from which to get netcdf files for inputs.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <ows:AllowedValues>
                        <ows:Value>bccaqv2</ows:Value>
                    </ows:AllowedValues>
                </LiteralData>
            </Input>
            <Input minOccurs="1" maxOccurs="1">
                <ows:Identifier>rcp</ows:Identifier>
                <ows:Title>RCP Scenario</ows:Title>
                <ows:Abstract>Representative Concentration Pathway
(RCP)</ows:Abstract>
                <LiteralData>
```

```
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <ows:AllowedValues>
                        <ows:Value>rcp26</ows:Value>
                        <ows:Value>rcp45</ows:Value>
                        <ows:Value>rcp85</ows:Value>
                    </ows:AllowedValues>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1000">
                <ows:Identifier>models</ows:Identifier>
                <ows:Title>Models to include in ensemble</ows:Title>
                <ows:Abstract>
When calculating the ensemble, include only these models. By default, all 24 models
are used.
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <ows:AllowedValues>
                        <ows:Value>24MODELS</ows:Value>
                        <ows:Value>PCIC12</ows:Value>
                        <ows:Value>BNU-ESM</ows:Value>
                        <ows:Value>CCSM4</ows:Value>
                        <ows:Value>CESM1-CAM5</ows:Value>
                        <ows:Value>CNRM-CM5</ows:Value>
                        <ows:Value>CSIRO-Mk3-6-0</ows:Value>
                        <ows:Value>CanESM2</ows:Value>
                        <ows:Value>FGOALS-g2</ows:Value>
                        <ows:Value>GFDL-CM3</ows:Value>
                        <ows:Value>GFDL-ESM2G</ows:Value>
                        <ows:Value>GFDL-ESM2M</ows:Value>
                        <ows:Value>HadGEM2-AO</ows:Value>
                        <ows:Value>HadGEM2-ES</ows:Value>
                        <ows:Value>IPSL-CM5A-LR</ows:Value>
                        <ows:Value>IPSL-CM5A-MR</ows:Value>
                        <ows:Value>MIROC-ESM-CHEM</ows:Value>
                        <ows:Value>MIROC-ESM</ows:Value>
                        <ows:Value>MIROC5</ows:Value>
                        <ows:Value>MPI-ESM-LR</ows:Value>
                        <ows:Value>MPI-ESM-MR</ows:Value>
                        <ows:Value>MRI-CGCM3</ows:Value>
                        <ows:Value>NorESM1-M</ows:Value>
                        <ows:Value>NorESM1-ME</ows:Value>
                        <ows:Value>bcc-csm1-1-m</ows:Value>
                        <ows:Value>bcc-csm1-1</ows:Value>
                    </ows:AllowedValues>
                    <DefaultValue>24MODELS</DefaultValue>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
```

```
                <ows:Identifier>thresh_tasmax</ows:Identifier>
                <ows:Title>Threshold</ows:Title>
                <ows:Abstract/>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <DefaultValue>UNSET 0 degC</DefaultValue>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>thresh_tasmin</ows:Identifier>
                <ows:Title>Threshold</ows:Title>
                <ows:Abstract/>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <DefaultValue>UNSET 0 degC</DefaultValue>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>freq</ows:Identifier>
                <ows:Title>Frequency</ows:Title>
                <ows:Abstract>Resampling frequency</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <ows:AllowedValues>
                        <ows:Value>YS</ows:Value>
                        <ows:Value>MS</ows:Value>
                        <ows:Value>QS-DEC</ows:Value>
                        <ows:Value>AS-JUL</ows:Value>
                    </ows:AllowedValues>
                    <DefaultValue>YS</DefaultValue>
                </LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>output_format</ows:Identifier>
                <ows:Title>Output format choice</ows:Title>
                <ows:Abstract>
Choose in which format you want to recieve the result
</ows:Abstract>
                <LiteralData>
                    <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-
2/#string">string</ows:DataType>
                    <ows:AllowedValues>
                        <ows:Value>netcdf</ows:Value>
                        <ows:Value>csv</ows:Value>
                    </ows:AllowedValues>
                    <DefaultValue>netcdf</DefaultValue>
                </LiteralData>
            </Input>
```

```xml
        </DataInputs>
        <ProcessOutputs>
            <Output>
                <ows:Identifier>output</ows:Identifier>
                <ows:Title>Result</ows:Title>
                <ows:Abstract>
The format depends on the 'output_format' input parameter.
</ows:Abstract>
                <ComplexOutput>
                    <Default>
                        <Format>
                            <MimeType>application/x-netcdf</MimeType>
                            <Encoding>base64</Encoding>
                        </Format>
                    </Default>
                    <Supported>
                        <Format>
                            <MimeType>application/x-netcdf</MimeType>
                            <Encoding>base64</Encoding>
                        </Format>
                        <Format>
                            <MimeType>application/zip</MimeType>
                            <Encoding>base64</Encoding>
                        </Format>
                    </Supported>
                </ComplexOutput>
            </Output>
            <Output>
                <ows:Identifier>output_log</ows:Identifier>
                <ows:Title>Logging information</ows:Title>
                <ows:Abstract>Collected logs during process run.</ows:Abstract>
                <ComplexOutput>
                    <Default>
                        <Format>
                            <MimeType>text/plain</MimeType>
                        </Format>
                    </Default>
                    <Supported>
                        <Format>
                            <MimeType>text/plain</MimeType>
                        </Format>
                    </Supported>
                </ComplexOutput>
            </Output>
        </ProcessOutputs>
    </ProcessDescription>
</wps:ProcessDescriptions>
```

Appendix C: Remote Sensing Processing Graph

Processing graph

```
<graph id="Graph">
    <version>1.0</version>
    <node id="Read">
        <operator>Read</operator>
        <sources/>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">

<file>/workspace/S2A_MSIL1C_20191212T155641_N0208_R054_T18TXR_20191212T191005.zip</fil
e>
        </parameters>
    </node>
    <node id="StackCreation">
        <operator>StackCreation</operator>
        <sources>
            <sourceProduct refid="Read"/>
            <sourceProduct.1 refid="Read(2)"/>
            <sourceProduct.2 refid="Read(3)"/>
            <sourceProduct.3 refid="Read(4)"/>
            <sourceProduct.4 refid="Read(5)"/>
        </sources>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement"/>
    </node>
    <node id="Read(2)">
        <operator>Read</operator>
        <sources/>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">

<file>/workspace/RS2_OK18072_PK188251_DK178156_F22_20090501_110525_HH_HV_SLC.zip</file
>
        </parameters>
    </node>
    <node id="Read(3)">
        <operator>Read</operator>
        <sources/>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">
            <file>/workspace/PROBAV_S1_TOA_X10Y02_20150617_1KM_V101.tif</file>
        </parameters>
    </node>
    <node id="Read(4)">
        <operator>Read</operator>
        <sources/>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">
            <file>/workspace/Deimos.tif</file>
        </parameters>
```

```xml
    </node>
    <node id="Read(5)">
        <operator>Read</operator>
        <sources/>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">

<file>/workspace/S1A_IW_GRDH_1SDV_20170519T224357_20170519T224422_016657_01BA57_B4F7.S
AFE.zip</file>
        </parameters>
    </node>
    <node id="Write">
        <operator>Write</operator>
        <sources>
            <sourceProduct refid="StackCreation"/>
        </sources>
        <parameters class="com.bc.ceres.binding.dom.XppDomElement">
            <file>/workspace/stacker_output.tif</file>
            <formatName>GeoTIFF</formatName>
        </parameters>
    </node>
</graph>
```

Appendix D: JSON File for the Climate Process Execute Request Body

JSON file for the climate process execute request body

```
{
   "inputs":[
      {
         "id":"lat",
         "data":
"53.21192218473093,53.21110066211657,53.30055393802849,53.208635999759395,53.290715033
438566"
      },
      {
         "id":"lon",
         "data":"-89.1980910629555,-89.12946274676571,-89.14181584367988,
-89.04436363469034,-89.04024593571896"
      },
      {
         "id":"start_date",
         "data":"1950"
      },
      {
         "id":"end_date",
         "data":"2100"
      },
      {
         "id":"ensemble_percentiles",
         "data":"10"
      },
      {
         "id":"dataset_name",
         "data":"bccaqv2"
      },
      {
         "id":"rcp",
         "data":"rcp26"
      },
      {
         "id":"models",
         "data":"PCIC12"
      },
      {
         "id":"freq",
         "data":"YS"
      },
      {
         "id":"output_format",
         "data":"netcdf"
      },
      {
         "id":"thresh_tasmax",
         "data":"0 degC"
      },
      {
         "id":"thresh_tasmin",
         "data":"-1 degC"
      }
   ],
   "response":"document",
   "mode":"auto",
   "outputs":[
      {
         "transmissionMode":"reference",
         "id":"output"
      }
   ],
}
```

Appendix E: CWL File for WPS 1.0 Provider

CWL file for the ice_days process of Finch WPS 1.0 provider

```
{
  "cwlVersion": "v1.0",
  "$namespaces": {
    "edam": "http://edamontology.org/"
  },
  "class": "CommandLineTool",
  "hints": {
    "WPS1Requirement": {
      "process": "ice_days",
      "provider": "https://finch.crim.ca/wps"
    }
  },
  "inputs": {
    "tasmax": {
      "default": {
        "mimeType": "application/x-netcdf",
        "schema": null,
        "encoding": "base64"
      },
      "type": {
        "items": "File",
        "type": "array"
      },
      "format": "edam:format_3650"
    },
    "freq": {
      "default": "YS",
      "type": {
        "symbols": [
          "YS",
          "MS",
          "QS-DEC",
          "AS-JUL"
        ],
        "type": "enum"
      }
    }
  },
  "outputs": {
    "output_netcdf": {
      "outputBinding": {
        "glob": "output_netcdf.nc"
      },
      "type": "File",
      "format": "edam:format_3650"
    },
      "output_log": {
        "outputBinding": {
          "glob": "output_log.*"
        },
        "type": "File",
        "format": "edam:format_1964"
      }
    }
  }
```

Appendix F: JSON Request Body to Execute WorkflowSubsetPicker

JSON request body to execute WorkflowSubsetPicker

```json
{
  "mode": "async",
  "response": "document",
  "inputs": [
    {
      "id": "resource",
      "href": "http://esgf-
data3.ceda.ac.uk/thredds/fileServer/esg_cmip6/CMIP6/ScenarioMIP/MOHC/HadGEM3-GC31-
LL/ssp245/r2i1p1f3/day/tasmax/gn/v20190906/tasmax_day_HadGEM3-GC31-
LL_ssp245_r2i1p1f3_gn_20150101-20201230.nc"
    },
    {
      "id": "lon0",
      "data": "-50"
    },
    {
      "id": "lon1",
      "data": "-80"
    },
    {
      "id": "lat0",
      "data": "40"
    },
    {
      "id": "lat1",
      "data": "60"
    },
    {
      "id": "metaindex",
      "data": 1
    }
  ],
  "outputs": [
    {
      "id": "output",
      "transmissionMode": "reference"
    }
  ]
}
```

Appendix G: Technical Interoperability Experiments

TIE #	Application name	EMS name	ADES name	Endpoint	Configuration	Issues or comments
1000	CRIM Stacker	CRIM Weaver	CRIM Weaver	CRIM	URL(R1,S1, S2)	Had to modify the app to work around working directory issues in SNAP
1001	CRIM Stacker	CRIM Weaver	CRIM Weaver	Boreal Cloud	URL(S1, S2)	Idem
1002	CRIM Stacker	CRIM Weaver	CRIM Weaver	EODMS	URL(S1, S2)	Idem
1003	CRIM Stacker	CRIM Weaver	CRIM Weaver	EODMS	LOCAL(S1, S2)	Idem
1004	CRIM Stacker	CRIM Weaver	CRIM Weaver	CRIM	URL(R1,Probav,Deimos)	Idem
1100	CRIM ML segmentation	CRIM Weaver	CRIM Weaver	CRIM	URL(S2)	https://ogc-ades.crim.ca/ADES/processes/TIE_crim_ml-segmentation/jobs/9933f190-1182-454f-848e-e69e1227dad0

TIE #	Application name	EMS name	ADES name	Endpoint	Configuration	Issues or comments
1101	CRIM ML segmentation	CRIM Weaver	Spacebel	Spacebel	URL(S2)	Provided CWL & docker definition for ML app. Error during deploy to Spacebel ADES. Expects a TB14-compliant AP like SatCen CWL [https://raw.githubusercontent.com/spacebel/tb/master/coherence-satcen.cwl]
1102	CRIM ML segmentation	CRIM Weaver	CRIM Weaver	EODMS	URL(S2)	N/A
1103	CRIM ML segmentation	CRIM Weaver	EOX	EOX	URL(S2)	N/A
1200	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	URL(CMIP6)	LLNL, Process subset wip : wps error
1201	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	URL(BCCAQv2)	Ouranos, Process subset wip : wps error
1202	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	URL(ANUSPLIN)	Ouranos, Process subset wip : wps error

TIE #	Application name	EMS name	ADES name	Endpoint	Configuration	Issues or comments
1203	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	URL(CMIP5) 2020-05-26	CEDA, Process subset wip : wps error
1204	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	URL(NRCAN-testdata)	Ouranos, Workflow Subset Picker, NRCAN test dataset wip : ems error
1205	CRIM climate services	CRIM Weaver	CRIM Weaver	CRIM	OpenSearch(CMIP6)	Using CEDA's directory
1301	CRIM non-docker test app	CRIM Weaver	CRIM Weaver	CRIM	LOCAL()	Tested on each weaver build, with travis-ci
1302	CRIM non-docker test app	CRIM Weaver	CRIM Weaver	Boreal Cloud	LOCAL()	Executed job can be found here [https://borealweb.nfis.org/ogc-pilot-ems/ems/processes/jsonarray2netcdf/jobs/b5fc6c8b-5997-4118-a1c1-a2cb96ec7be9]
1303	CRIM non-docker test app	CRIM Weaver	CRIM Weaver	EODMS	LOCAL()	N/A

TIE #	Application name	EMS name	ADES name	Endpoint	Configuration	Issues or comments
2000	Eu SatCen interferometry	CRIM Weaver	CRIM Weaver	CRIM	URL	CWL with directories disallowed. URL causes MemoryError at stage-in/execute with tmp user/password
3000	Pixalytics mosaicing	CRIM Weaver	CRIM Weaver	CRIM	URL	N/A
3001	Pixalytics hello world	CRIM Weaver	CRIM Weaver	CRIM	URL	N/A
4000	ESA mosaicing - Rhea	CRIM Weaver	CRIM Weaver	CRIM	URL	Executed job can be found here [https://ogc-ades.crim.ca/ADES/processes/TIE_rhea_mosaicing/jobs/093d6308-561e-4b12-aa4b-081d420071fa]

Appendix H: Example of Configurations for S3 Buckets Support

File ~/.aws/credentials

```
[default]
aws_access_key_id = <...>
aws_secret_access_key = <...>
```

File ~/.aws/config

```
[default]
region = ca-central-1
output = text
color = on

[profile eodms]
role_arn = arn:aws:iam::852057203659:role/EOPilotSSM
source_profile = default
region = ca-central-1
```

Appendix I: Revision History

Table 9. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
May 20th, 2020	T. Landry	.1	all	initial version
June 2nd, 2020	T. Landry	.2	all	complete TOC
June 6th, 2020	F. Charette-Migneault	.3	all	docs for platforms
June 7th, 2020	F. Charette-Migneault	.4	all	docs for applications
June 8th, 2020	M. Beaulieu	.5	application	docs for remote sensing app
June 12th, 2020	M. Provencher	.6	application	docs for climate services
June 15th, 2020	F. Charette-Migneault	.7	all	TIES
June 23rd, 2020	T. Landry	.8	all	initial documentation dump
July 28th, 2020	T. Landry, D. Byrns, F. Charette-Migneault	.9	platforms, apps	structure and details
July 31th, 2020	T. Landry, F. Charette-Migneault	.95	summary, eo apps	filled most information
August 5th, 2020	T. Landry	1.0	all	editorial changes, final draft
October 1st, 2020	T. Landry, F. Charette-Migneault	1.1	Summary, conclusion, EO platforms, EO apps	EODMS and PBC findings, updates
October 8th, 2020	T. Landry	1.1.1	EO platforms	Adding NRCAN intentions, adding NRCAN contributors
October 23rd, 2020	G. Hobona	1.1.2	all	final staff review

Appendix J: Bibliography

[1] Earth System Grid Federation (ESGF) Compute Challenge. OGC 19-003,Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2019).

[2] Sacramento, P., others: OGC Testbed-14: Application Package Engineering Report. OGC 18-049,Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2018).

[3] Sacramento, P., others: OGC Testbed-14: ADES & EMS Results and Best Practices Engineering Report. OGC 18-050,Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2018).

[4] Landry, T., others: OGC Testbed-14: Machine Learning Engineering Report. OGC 18-038,Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2018).

[5] Meek, S., others: OGC Testbed-15: Machine Learning Engineering Report. OGC 19-027r2,Open Geospatial Consortium, http://www.opengeospatial.org/docs/er (2018).

[6] Chen, C., others: OGC Testbed-13: Cloud ER. OGC 17-035,Open Geospatial Consortium, http://docs.opengeospatial.org/per/17-035.html (2017).

[7] Zuhlke, M., Fomferra, N., Brockmann, C., Peters, M., Veci, L., Malik, J., Regner, P.: SNAP (sentinel application platform) and the ESA sentinel 3 toolbox. ESASP. 734, 21 (2015).

[8] Huang, X., Zhang, L., Li, P.: Classification and Extraction of Spatial Features in Urban Areas Using High-Resolution Multispectral Imagery. Geoscience and Remote Sensing Letters, IEEE. 4, 260–264 (2007).

[9] Otsu, N.: A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics. 9, 62–66 (1979).

[10] Helber, P., Bischke, B., Dengel, A., Borth, D.: EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification, (2017).