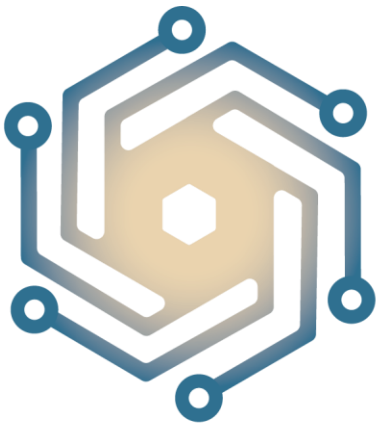

CLINT

CLIMATE INTELLIGENCE

D8.5

CLIMATE RESILIENCE INFORMATION SYSTEMS ARCHITECTURE

June 2024

This project is part of the H2020 Programme supported by the European Union, having received funding from it under Grant Agreement No 101003876

Programme Call:	Building a low-carbon, climate resilient future: climate action in support of the Paris Agreement (H2020-LC-CLA-2018-2019-2020)
Grant agreement ID:	101003876
Project Title:	CLINT
Partners:	POLIMI (Project Coordinator), CMCC, HEREON, CSIC, SMHI, HKV, E3M, TCDF, DKRZ, IHE, ECMWF, UAH, JLU, OGC, UCM
Work-Package:	WP8
Deliverable #:	D8.5
Deliverable Type:	Other
Contractual Date of Delivery:	30 June 2024
Actual Date of Delivery:	30 June 2024
Title of Document:	Climate Services Information Systems Architecture
Responsible partner:	DKRZ
Author(s):	Nils Hempelmann (OGC)
Content of this report:	Detailed description of backend related software architecture. The report from T8.3 integrates milestone MS21 and will include the descriptions new developed and already available components to establish CSIS services providing ML codes as processes. The report will also include the appropriate code repositories.
Availability:	Public

Document revisions		
Author	Revision content	Date
Nils Hempelmann (OGC)	First Draft based on MS21 report	Nov 2023
Nils Hempelmann (OGC)	Update of Introduction and description text	May 2024
Leone Cavicchia (CMCC), Marcello Restelli (POLIMI)	Reviews	June 2024
Guido Ascenso (POLIMI), Andrea Castelletti (POLIMI)	Final revisions	25/06/2024

Table of Content

LIST OF ACRONYMS

Abbreviations

AB: Advisory Board
AI: Artificial Intelligence
CA: Consortium Agreement
CDS Climate Data Store
CLINT: Climate Intelligence
CS: Climate Service
C3S: Copernicus Climate Change Service
DCV: Data Catalogue Vocabulary
DKRZ German Climate Computing Center
DMP: Data Management Plan
DoA: Description of Action (Annex I of the Grant Agreement)
DOI: Digital Object Identifier
EC: European Commission
ECMWF: European Centre for Medium-Range Weather Forecasts
FAIR: Findable, Accessible, Interoperable, Reusable
GA: Grant Agreement
GDPR: General Data Protection Regulation
ML: Machine Learning
NetCDF: Climate Data Format
PC: Project Coordinator
PFB: Publish->Find->Bind
ORD Open Research Data
SMHI: Swedish Meteorological and Hydrological Institute
TCDF: The Climate Data Factory
USGS: United States Geological Survey
WDCC: World Data Center for Climate
WPx: Work Package (where x is the WP number)
WEF: Water-Energy-Food

EXECUTIVE SUMMARY

The following is the first draft description of the architecture of the CLINT Climate Resilience Information System, which is hosted and maintained at the DKRZ high-performance environment as a DEMO service. At the current state, six prototypes of climate resilience application packages have been deployed, forming the CLINT Climate Resilience Information Systems. The DEMO service provides technical services to detect extreme weather events based on scientific algorithms developed by the CLINT consortium. The deployed application packages are running with AI components. This deliverable lays out the DEMO service architecture itself, as well as the blueprint of the workflow on how to set up a climate application package and how to deploy it into the architecture. The entire architecture is following the FAIR principles to ensure FAIR Climate Service as described in the CLINT Data Management Plan. The CLINT DEMO service is envisioned to be deployed in the European digital infrastructure, namely the Copernicus Climate Change Service C3S with the underpinned Climate and Atmospheric Data Store (CADS) but also Wekeo or the European Open Science Cloud (EOCS). Deployed application packages are described with their scientific functionality in detail in the MS28 and D8.7 *'Extreme Events ML in Climate Services Information Systems'*. The backend of this DEMO system is deployed at the German Climate Computing Center (DKRZ), but is deployable in all kinds of high-performance environments.

1 INTRODUCTION

Since the upcoming challenge of climate change is on a global dimension, it has, therefore, to be addressed on a global scale. The interoperability of data, information, and knowledge, as well as all ready-to-use application packages, is essential to increase the efficiency of knowledge generation. Therefore, the FAIR principles play a special role in designing Climate Resilience Information Systems (CRIS). Not only does data need to be findable, accessible, interoperable, and reusable, but the entire climate information ecosystem around the data should adhere to the same principles. FAIR climate services are optimising the efficiency of available budget, resources, or access to knowledge. In Figure 1, FAIR Principles are schematically shown with the options on how to realise them. Applying FAIR principles to CRIS is here understood in the sense of:

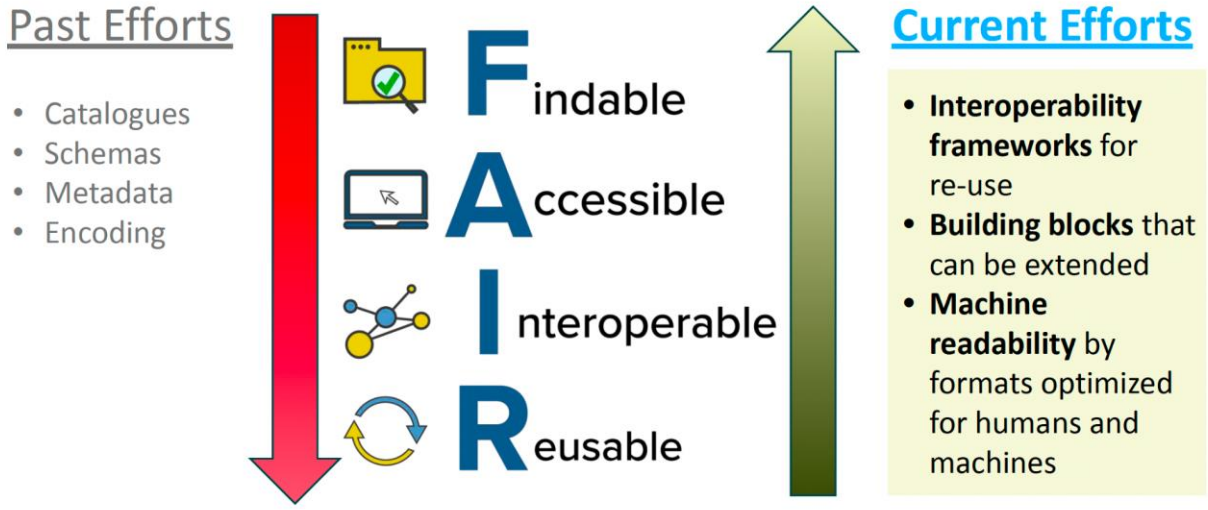

Figure 1: Classic (past) efforts and modern (current) efforts to enable FAIR principles for data.

Findable	Besides data, services need to be Findable within federated data distribution architectures and public catalogs of well-documented analytical processes.
Accessible	Remote storage and CPU should be operationally Accessible to all including low bandwidth regions and closing digital gaps to 'Leave No One Behind'.
Interoperable	Data inputs, outputs and processing API standards are the necessary conditions to ensure the system is Interoperable.

Reusable	Reusable components can be realised by modular architectures with swappable components, data provenance systems, and rich metadata.

The need to report information for local, national, or international climate policy frames is constantly increasing. FAIR climate services are playing an important role in this context. Therefore, well-designed architectures of the technical backends for FAIR Climate Service for reliable climate-related location information are essential to establish climate change-related policy strategies and their climate action plans, which need to be elaborated on, improved, and realised.

The architecture of the AI-enhanced CLINT-CRIS built within the CLINT project follows the FAIR Climate Services' ideas to move towards a global interoperable CRIS. This deliverable D8.5 is built upon the MS21 report and describes the current state of the CRIS architecture and how to integrate the AI compartments into it. It is structured in the following chapters:

- Architecture and Functionality of the DEMO Service
- Transfer a scientific method into technical service
- Deployment of climate application packages

This deliverable is accomplished by the Data management plan (D8.6/D8.8) and the descriptions of the application packages on *'Extreme Events ML in Climate Services Information Systems'* (D8.7). Most of the text published in this document can also be found online within the CLINT open source repositories online documentations or at the OGC websites. Furthermore, training material has been produced for the CLINT summer school and will be disseminated and enhanced within the project live time. There are descriptions on how to build application packages which are FAIR climate services compliant. Here, we understand climate application packages for technical climate services as standalone software in line with the OGC API standards.

2 THE ARCHITECTURE OF THE CLINT DEMO SERVER

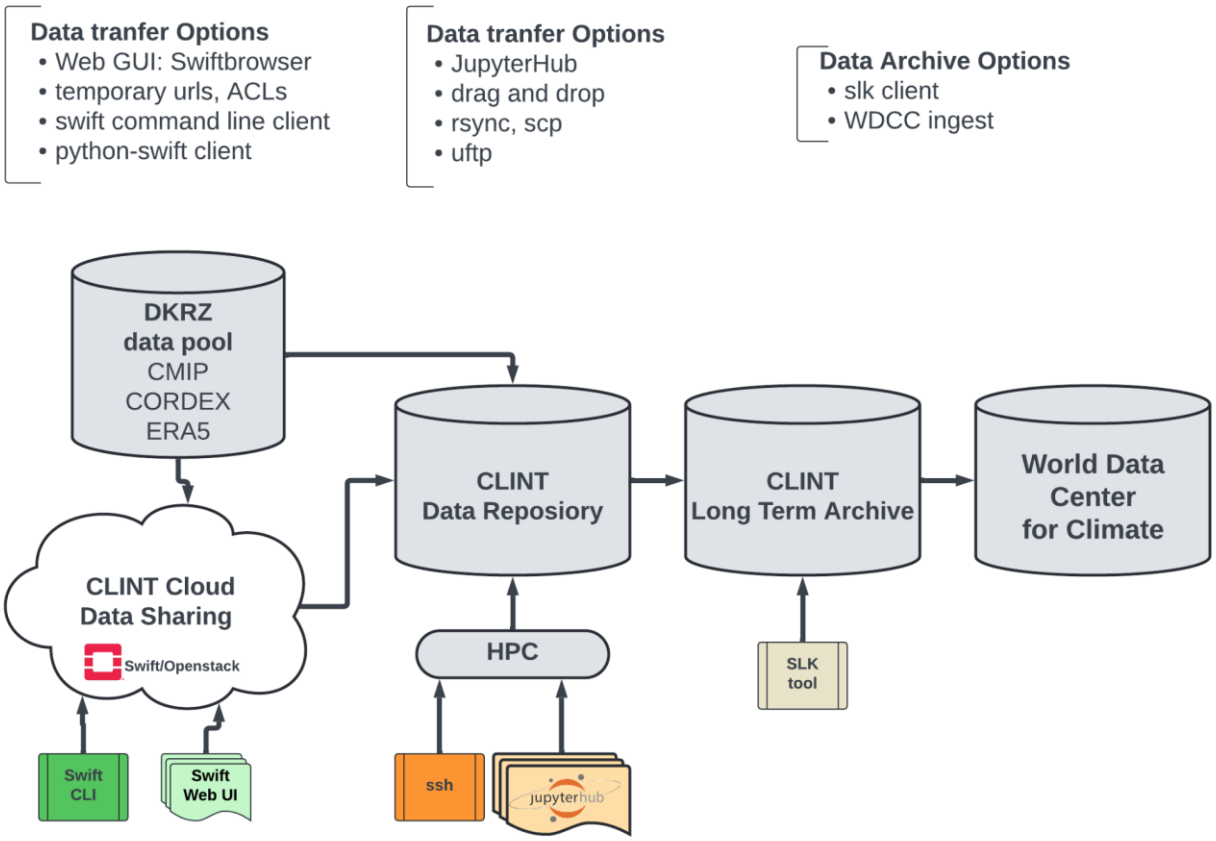

Figure 2: General overview of the CRIS CLINT processing service next to the data repository.

The CLINT DEMO of the established prototypes is deployed at DKRZ[1]. It is deployed close to the data repository on virtual machines (VMs): VMs for the AI-enhanced WPS (e.g., Duck) and a separate VM for the GUI (called Phoenix) that allows to interact with the WPS (See Figure 3)

[1] DEMO of the CLINT CRIS architecture realisation: https://clint.dkrz.de/

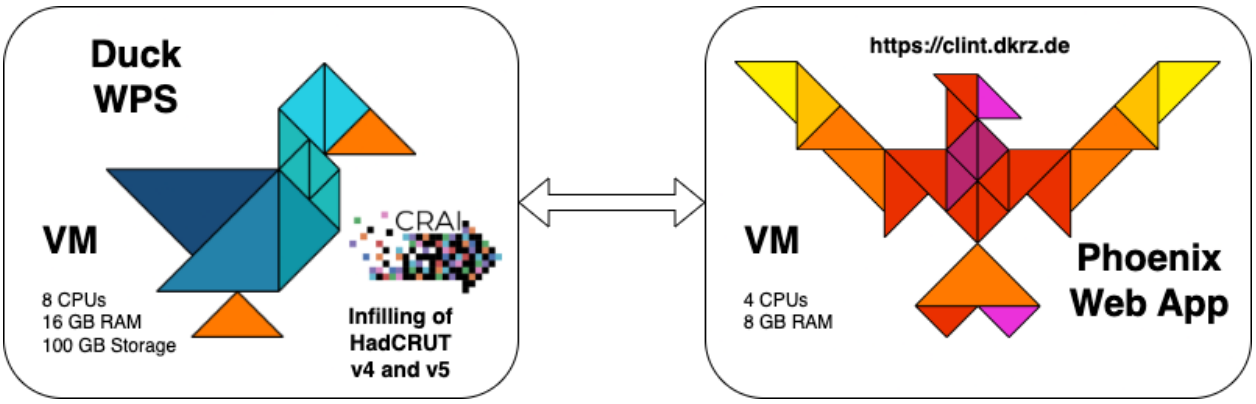

Figure 3: Demo deployment of the Duck WPS (infilling algorithm) and the Phoenix Web App.

As the first implemented prototype on inpainting missing values, the climate application package Duck is deployed into the architecture of the CLINT CRIS.

The Duck WPS is created from a cookiecutter template for a WPS processing service (PyWPS). It provides the CRAI software (https://github.com/FREVA-CLINT/climatereconstructionAI) to infill HadCRUT4 and HadCRUT5 data from MetOffice using a trained AI model (see Deliverable D2.2) as a web service. The Duck WPS is deployed on a VM at DKRZ cloud infrastructure with enough resources to run 2 infilling processes in parallel (8 CPUs, 16 GB RAM). A Slurm queue schedules the processing jobs in case more jobs are in-coming from user requests. The Duck WPS is not accessed by users directly. There is the Phoenix Web App, which provides a friendly user interface to the users to submit jobs and see the outputs. This Web App is installed on an additional VM at the DKRZ cloud infrastructure. The app can be accessed by the URL: clint.dkrz.de

3 CREATE A CLIMATE APPLICATION PACKAGE

The climate application packages built for the CLINT architecture follow the FAIR climate services ideas, described in the introduction. To ensure this, OGC standards are being used. They are Web Processing Services respectively the related OGC API Processes.

Web Processing Service and OGC API Processes

The Open Geospatial Consortium (OGC) provides standards, such as APIs for web services to access and process data. In CLINT, we are using the Web Processing Service API with the Python implementation of PyWPS. The WPS standard allows running processing tools as a service via a simple Web API. The latest API for processing is OGC API Processes, which uses a REST-based API. PyWPS also includes an implementation of OGC API processes.

The following sections describe how to transfer scientific methods into technical services which are deployable in climate resilience information systems.

Get started with the environment

The climate building blocks (=birds) are designed to run in their own environment. This strategy permits to avoid dependency conflicts, when multiple birds are installed in a Climate Services Information System. The environments are organised with the conda package manager.

Set up the skeleton of your new climate building block

The following sections describe how to transfer scientific methods into technical services which are deployable in climate resilience information systems. This is necessary when appropriate scientific methods are not already available and accessible as technical services. Birdhouse components lower the barriers to setting up new Climate building blocks by providing tools that enable you to build your own customised OGC API Processes application in support of web-based geospatial (climate) data analysis.

Within the Birdhouse organisation, you can find:

- A Cookiecutter template to create your own PyWPS compute service.

The cockiecutter can be installed with:

conda install -c conda-forge cookiecutter cruft

Example: build the application package 'duck'

Here is an example of how to build your processing service application, *duck*, using the cookiecutter template. Run the cookiecutter with the birdhouse template:

cruft create https://github.com/bird-house/cookiecutter-birdhouse.git

Once cookiecutter clones the template, you will be asked a series of questions related to your project:

full_name [Full Name]: Alice Kingsleigh

email [your@email]: alice@wonderland.org

github_username [bird-house]: clint

project_name [Babybird]: duck

project_slug [duck]:

project_repo_name [duck]:

project_readthedocs_name [duck]:

project_short_description [A Web Processing Service for Climate Data Analysis.]: A Demo Web Service for Clint.

version [0.1.0]:

Select open_source_license:

1 - Apache Software License 2.0

2 - MIT license

3 - BSD license

4 - ISC license

5 - GNU General Public License v3

6 - Not open source

Choose from 1, 2, 3, 4, 5, 6 [1]: 1

http_port [5000]:

use_pytest [y]: y

create_author_file [y]: y

We have created a *duck* app for the *CLINT* project using this template.

Create the birds environment

The new bird is coming as a fully operational service. The appropriate environment is defined in the ./{birdname}/environment.yml file and can be installed with

mamba env create

or

conda env create

to switch into the environment:

conda activate duck

Installing and running the bird

Since the climate building blocks are web services, they need to be started to make the service available. They can be installed in the activated environment of the bird:

pip install -e .

This is installing the service using pip/python. When running the command `duck start`, the service is up and running.

You can test the service in a web browser by entering the following URLs:

- http://localhost:5000/wps?version=1.0.0&request=GetCapabilities&&service=WPS
- http://localhost:5000/wps?version=1.0.0&request=DescribeProcess&&service=WPS&identifier=hello

Fill your new climate building block with scientific algorithms

Once your new Climate Building block is created, it needs to be filled with scientific algorithms. In contrast to a script that you are running locally in an individual way, a service needs to be standardised. The following figure illustrates the basic design of a technical service. Input data needs to be provided by a given URL to the appropriate resources and in addition parameters to modify the execution. On the other side, you need to define the output files.

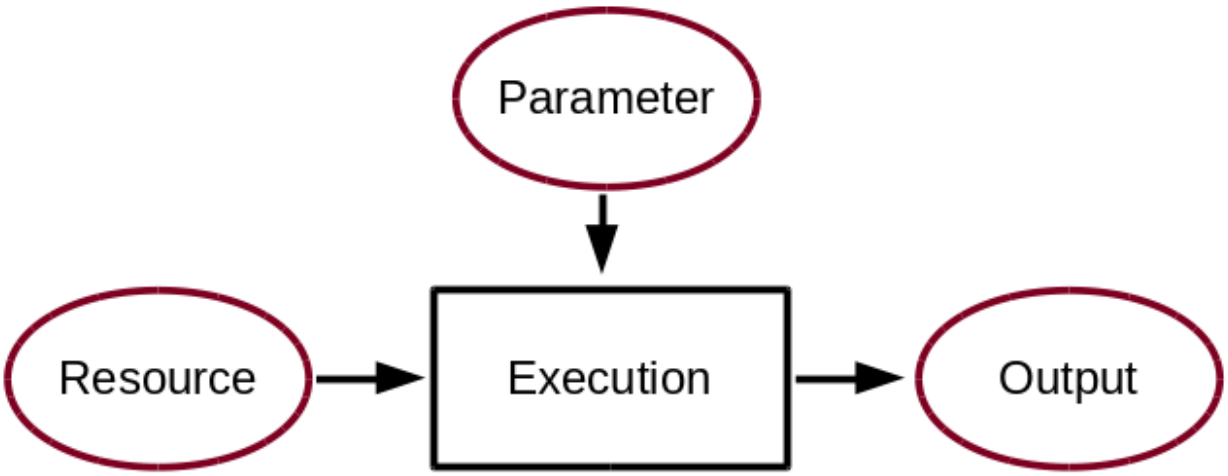

Figure 4: the principal concept of a processing service.

Basically, your existing code can be integrated by simply taking care of some rules:

- no hard coding of path to local data sources since the building block should run on all kinds of different servers
- encapsulating the processing steps in `try` and `exception` brackets with sensible log messages

Writing functions

A Process is calling several functions during the execution. Since WPS is an autonom running process several eventualities need to be taken into account. If irregularities are occurring, it is a question of the process design if the performance should stop and return an error or continue with a modified result.

In practice, the functions should be encapsulated in `try` and `except` calls and appropriate information given to the logfile or shown as a status message. The logger has several options to influence the running code and the information written to the logfile:

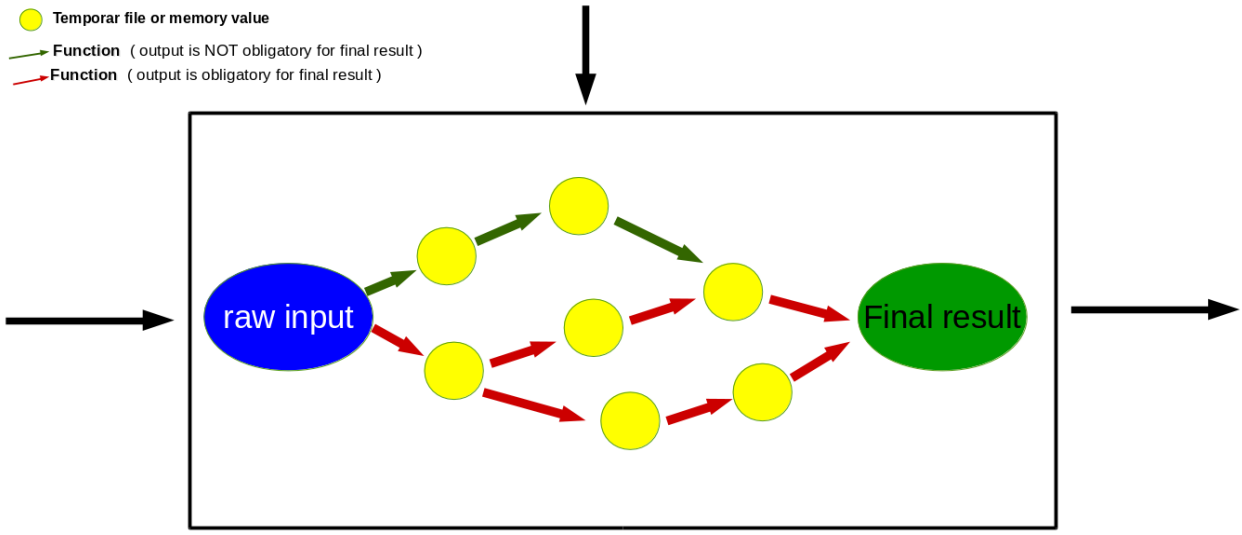

Figure 5: General design of a technical processing service, which can run autonomously as a technical climate service.

```
# the following two lines need to be at the beginning of the *.py file.

# The ._handler will find the appropriate logfile and include timestemps

# and module information into the log.

  import logging

  LOGGER = logging.getLogger("PYWPS")



  # set a status message

  per = 5  # 5 will be 5% in the status line

  response.update_status('execution started at : {}'.fromat(dt.now()), per)

  try:
```

```
        response.update_status('the process is doing something: {}'.fromat(dt.now()),10)
        result = 42
        LOGGER.info('found the answer of life')
    except Exception as ex:
        msg = 'This failed but is obligatory for the output. The process stops now, because: {} '.format(ex)
        LOGGER.error(msg)
    try:
        response.update_status('the process is doing something else : {}'.fromat(dt.now()), 20)
        interesting = True
        LOGGER.info(' Thanks for reading the guidelines ')
        LOGGER.debug(' I need to know some details of the process: {} '.format(ex))
    except Exception as ex:
        msg = 'This failed but is not obligatory for the output. The process will continue. Reason for the failure: {} '.format(ex)
        LOGGER.exception(msg)
```

4 DEPLOY A CLIMATE APPLICATION PACKAGE AS TEHCNICAL CLIMATE SERVICE

Example: Duck Demo App

Duck is a demo web-application of AI-enhanced Climate Science. It has the Phoenix web-application from the Birdhouse collection as a user web frontend. It makes use of the PyWPS Python package, which is an implementation of the Web Processing Service standard from the Open Geospatial Consortium, to execute the AI-enhanced algorithms as a service.

Duck uses CRAI, a state-of-the-art deep learning-based inpainting technology, to infill missing values in climate datasets.

The current demo gives the possibility to infill near-surface air temperature anomalies in the HadCRUT4 and HadCRUT5 datasets. The input and output netCDF files are handled through an intuitive user interface, Phoenix.

The current demo gives the possibility to infill near-surface air temperature anomalies in the HadCRUT4 and HadCRUT5 datasets.

Install a climate application package with conda/mamba

We need mamba to install the requirements. Use your existing mamba or install it from here:

https://github.com/conda-forge/miniforge

Get the source:

git clone https://github.com/climateintelligence/duck.git

cd duck

Create the conda environment for duck:

mamba env create

Activate the duck environment:

conda activate duck

Install duck in this environment:

pip install -e .

Start your duck as a web service:

duck start -d

Check if it is responding by sending a WPS GetCapabilities request:

http://localhost:5000?service=WPS&request=GetCapabilities

You can stop the service with:

duck stop

CLIMATE RESILIENCE INFORMATION SYSTEMS ARCHITECTURE

Overview of the deployed application packages

Within tasks 8.4.1 and 8.4.2 two application packages are supposed to be developed. Since WP8 is far ahead schedule it was possible to develop more than two packages with high quality of the deployed scientific algorithm which is implemented in the package. Software repositories and full online documentation can be found here:

- CLINT GitHub repository. https://github.com/climateintelligence/
- General Documentation - Smartduck: Climate application packages documentation

And and overview of the application packages (table 1) are described in the Milestone report for MS28.

Table 1: CLINT Climate Resilience Application Packages.

Package	Topic	Documentation	GitHub repository	Deployed
Duck	Inpainting Missing Values	clint-duck	duck	clint.dkrz.de
Owl	Heatwaves and Warm nights	clint-owl	owl	clint.dkrz.de
Albatross	Drought vulnerability	clint-albatross	albatross	clint.dkrz.de
Shearwater	Tropical Cyclone Genesis	shearwater	shearwater	clint.dkrz.de
Dipper	Flood, Hydrological Models	clint-dipper	dipper	clint.dkrz.de
Hawk	Causality analyses	clint-hawk	hawk	clint.dkrz.de
Phoenix	Graphical User Interface	pyramid-phoenix	bird-house/pyramid-phoenix	clint.dkrz.de

CLINT

CLIMATE INTELLIGENCE

CLIMATE RESILIENCE INFORMATION SYSTEMS ARCHITECTURE