Project Title:	Baltic+ BalticAIMS
Document Title:	D3.2 System and Service Chain Readiness Report SRR
Version:	1.1 Public version
Author(s) and affiliation(s):	Martin Böttcher, Carole Lebreton, Carsten Brockmann, Uwe Lange, BC Mikko Kervinen, Sampsa Koponen, Hanna Alasalmi, Jenni Attila, Vesa Keto SYKE
Version history:	0.1 10.2.2022. Template 1.0 24.3.2022 Version submitted for SRR review 1.1 06.04.2022 Public version
Distribution:	ESA, Project team, public

Contents

Glossary

API	Application programming interface
EO	Earth Observation
GIS	Geographic Information System
GUI	Graphic User Interface
HELCOM	Helsinki Commission
OGC	Open Geospatial Consortium
VM	Virtual Machine
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Mapping Service
WMTS	Web Mapping Tile Service

1 Introduction

1.1 Purpose and scope

This document is the service chain readiness report for the BalticAIMS services. This report focusses on the system setup and configuration and describes the installation status reached.

This document is complemented by the BalticAIMS service chain verification report [SVR, D3.1] which describes how the system is used, how to add data and how to access it from clients. The two documents are the response to the three WP 2 documents Service portfolio definition [Portfolio], Data and platform provisioning plan [Platform], and Service delivery chain specification [ChainSpec].

1.2 Document overview

After this formal introduction

section 2 provides an overview of showcases and datasets, services, and system elements that implement them

section 3 describes how the infrastructure has been set up and which structures have been defined for user accounts, directories on the VM and in cloud storage, the domain configuration, and configuration management

section 4 describes how the services for xcube, geodb, JupyterHub, and TARKKA are set up and configured for BalticAIMS

2 Overview

This section provides an overview of showcases and datasets, services, and system elements that implement them.

2.1 Showcases and datasets

The 5 show cases are

- A: Provide EO based information to be used in user legacy systems for spatial planning
- B: Monitor the effects of nutrient flow from the drainage basin to the coastal waters
- C: Monitoring the impacts of coastal activities
- D: Combination of Coastal Zone mapping and CMEMS coastal water quality material
- E: Monitoring of temperature anomalies

Each show case is further elaborated in several user stories with concrete applications. Details are provided in D2.1.

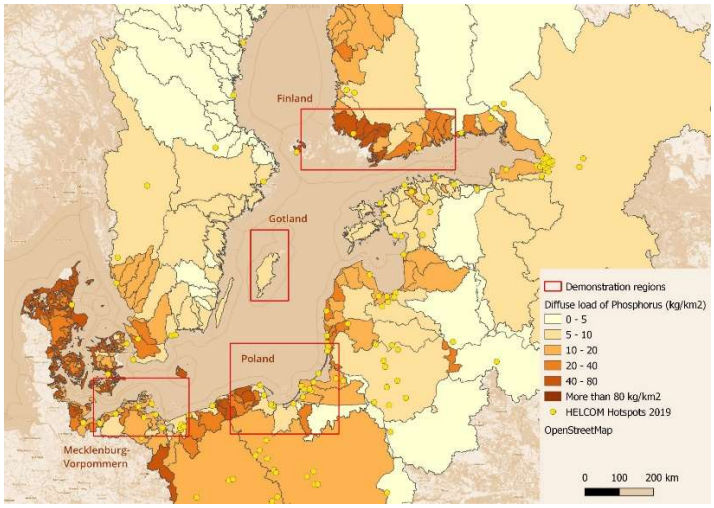

Figure 2-1: BalticAIMS showcase areas

There are 4 areas the show cases will be applied to in different combinations:

- 1. Archipelago Sea, Finland, and Helsinki
- 2. Gotland
- 3. Mecklenburg-Vorpommern
- 4. Poland

Datasets for the different showcases and areas are either gridded (time series of) image data, or tabular (series of) feature data. They are collected in BalticAIMS, sometimes processed or post-processed for BalticAIMS, and sometimes collected from other sources and made available in a harmonised analysis-ready way in BalticAIMS.

- Examples of gridded data are the high-resolution water quality datasets of the central Baltic Sea systematically generated by SYKE, or the Corine Land Cover classes provided at 4 time steps. Gridded data is made available in the xcube service.
- Examples of feature data are the N and P load from agriculture made available by HELCOM, or the coastal water body catchments provided by SMHI. Tabular feature data is made available in the geodb service.

2.2 Services and system elements

The BalticAIMS services used by the different show cases are:

- Showcase A makes available EO data for spatial planning using the user systems, mainly GIS. The GIS will read from BalticAIMS data interfaces, in particular WCS and WFS, to access data cubes and GeoDB. Simple static files are served from BalticAIMS geo file server as well.
- Showcase B provides data to analyse the effects of nutrient flow from drainage basins to coastal water. Main service used is TARKKA. The showcase may also use the OGC services provided by BalticAIMS.

- Showcase C provides data to analyse the impact of coastal activities. TARKKA demonstrates the service. GIS interfaces may be used as well.
- Showcase D maps coastal zones. It uses the BalticAIMS data service to include selected layers into the analysis.
- Showcase E makes available EO data to analyse temperature anomalies. It mainly uses TARKKA to demonstrate the service.

BalticAIMS data services are based on TARKKA, xcube, and geodb. The involved elements are shown in Figure 2-2.

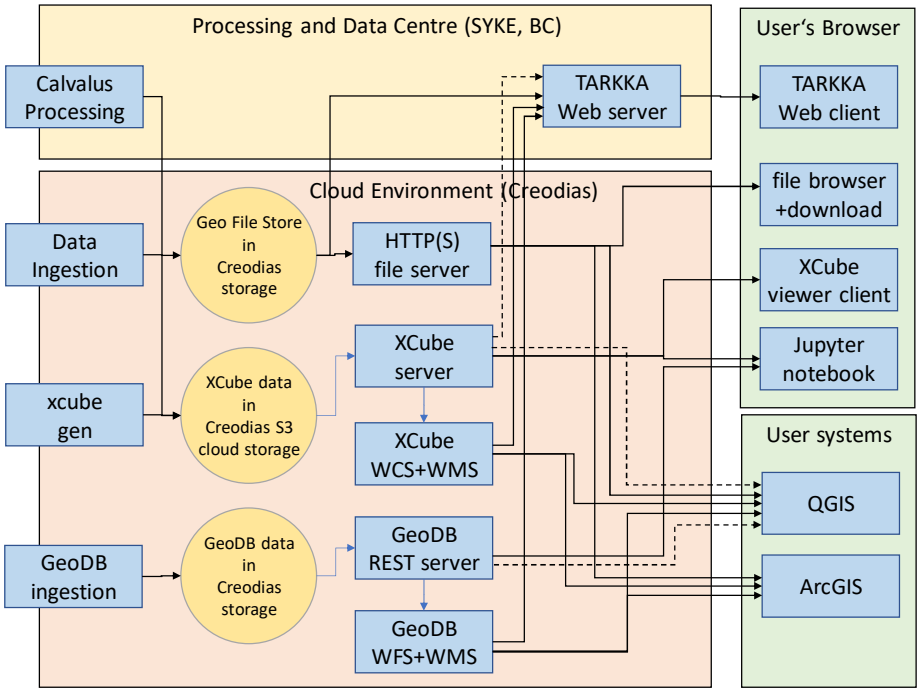

Figure 2-2: BalticAIMS system elements

- Raster time series data is either ingested and prepared with xcube gen, or it is processed by BalticAIMS and stored in XCube zarr format. This data is accessed via XCube server and viewer or via OGC WMS or WCS either by TARKKA or by user GIS applications.
- Feature data is inserted into GeoDB and served via OGC WFS either to TARKKA or to user GIS applications.
- Unstructured or simple file data can also be stored in a structured geo file store and served via HTTP for download or direct access by user GIS applications.

3 System structures and configuration

This section describes how the infrastructure has been set up and which structures have been defined for user accounts, directories on the VM and in cloud storage, the domain configuration, and configuration management.

3.1 Infrastructure and VM setup

The infrastructure on Creodias comprises a VM, means of access like network, and cloud storage. There is an account of SYKE on Creodias that owns the infrastructure, and one personal account of Brockmann Consult to access and work with the infrastructure. The actual setup has been done manually using the Openstack GUI. The script parts below use openstack and S3 to define what has been set up.

```
# download rc3 file from openstack Web interface upper left menu of Creodias
# store as dot-openrc-balticaims.sh
. dot-openrc-balticaims.sh

# create EC2 credentials and configure S3 access
openstack ec2 credentials create
s3cmd --configure
# use access and secret, Region1, s3.waw2-1.cloudferro.com
mv ~/.s3cfg dot-s3cmd-balticaims

# create and upload ssh key
ssh-keygen -t rsa -f balticaims_key.priv
chmod 600 balticaims_key.priv
mv balticaims_key.priv.pub balticaims_key.pub
openstack keypair create --public-key balticaims_key.pub

# create rule for restricted network access from BC and from SYKE
openstack security group create ba
openstack security group rule create --ingress --remote-ip 217.110.123.42 --dst-port 1:65535 ba
openstack security group rule create --ingress --remote-ip 217.110.123.43 --dst-port 1:65535 ba
openstack security group rule create --ingress --remote-ip 193.66.21.0/24 --dst-port 1:65535 ba
openstack security group rule create --ingress --remote-ip 192.58.53.0/24 --dst-port 1:65535 ba
# ?or is it openstack security group rule create --ingress --remote-ip 193.66.21 --dst-port 1:65535 ba

# create bucket
s3cmd -c dot-s3cfg-balticaims s3://balticaims

# create VM
openstack image list
openstack flavor list
openstack network list
openstack server create --image 45190492-e881-4543-85a2-15a2e525cb82 --flavor hm.large --
network private_network_28927 --network eodata --nic net-id=private_network_28927,v4-fixed-
ip=192.168.0.2 --security-group default --security-group ba --key-name balticaims balticcube

# create floating IP
openstack floating ip create external
openstack server add floating ip balticcube 185.178.84.146
```

Figure 3-1: Scripted setup of the BalticAIMS infrastructure with VM, network, and S3 bucket in Creodias

- There is an openstack configuration file that can be downloaded from the Creodias Openstack GUI when being logged in with the BC user account. Sourcing this configuration prompts for the password. Then, openstack commands can be run against Creodias.

- S3 cloud storage credentials are generated with an openstack command. The result is inserted into a s3cmd configuration. With it, s3cmd commands can be run against Creodias.
- An SSH key pair for balticaims is registered with Creodias. It will be injected into VMs being created. The private key is required to login into the VM created later.
- A security group is created for the private network on Creodias. It allows access to the VM from IP addresses of SYKE and BC. Additional rules will be added later for the inverse proxy to provide access to BalticAIMS services for everyone.
- An S3 cloud storage bucket called balticaims is created.
- A VM with 32 GB RAM, 4 cores, 120 GB local disk space is created. ssh key, security group, and VM name are provided as parameters.
- A new floating IP is created and linked to the VM.

```
# login to the VM
ssh -i balticaims_key.priv eouser@185.178.84.146

# entry into my local /etc/hosts
185.178.84.146  balticcube

# entry into my private ~/.ssh/config
Host balticcube
     User eouser
     IdentityFile /home/martin/tmp/dias-inst/balticaims_key.priv

# check for S3 access
s3cmd -c dot-s3cfg-balticaims ls
```

Figure 3-2: Verification and local setup of access to the infrastructure

- Login to the balticcube VM is possible as standard user eouser.
- With an entry in /etc/hosts and an entry in ~/.ssh/config login can be simplified.
- Access to the S3 bucket is possible as well.

3.2 System user accounts

There are user accounts used with the different services (Table 3-1).

Table 3-1: BalticAIMS system user accounts

Account	Domain and purpose
eouser	Unix ssh user of balticcube VM, sudoer
baadmin	geodb user, postgres user, JupyterHub user, allowed to add/delete collections
bauser	geodb user, postgres user, JupyterHub user, common user to read data
jupyter-baadmin	Unix account with notebooks and hidden credentials for geodb
jupyter-bauser	Unix account with notebooks and hidden credentials for geodb
postgres	Postgresql administrator user
authenticator	Postgresql lightweight user with rights for impersonation
cloaky	Postgresql user, owner of the Keycloak configuration in the database
martin	Keycloak admin user
admin	Geoserver admin user

More specific users in addition to bauser can be added if specific data shall be visible for a certain user.

3.3 System directory structures

The two directory structures on the VM Posix file system (local disk) and on the S3 cloud storage provide dedicated places for software, configuration, and data used by the services.

Table 3-2: VM directory structure for software, configuration, and data

Path	Content
/balticaims	root directory of the VM data partition
/balticaims/software	root directory of installed software packages for services
/balticaims/software/miniconda3-ba	Anaconda Python installation for xcube and geodb
/balticaims/software/xcube	xcube software installation
/balticaims/software/xcube-viewer	viewer software installation
/balticaims/software/node-v14.17.0-linux-x64	viewer Javascript software
/balticaims/software/xcube-geodb	xcube geodb software installation
/balticaims/software/postgrest	postgrest executable and configuration
/balticaims/software/keycloak-16.1.0	Keycloak software
/balticaims/software/geoserver-2.19.4	Geoserver instance installation
/balticaims/software/xcube-gen-balticaims	Plugins for xcube gen, maintained on github
/balticaims/packages	attic for downloaded software package versions before installation and configuration
/balticaims/www	root directory for index.html static web page and for xcube viewer configuration
/balticaims/data	root directory for geo file server data
/balticaims/data/rasterdata	gridded data, structured by collection, region, time
/balticaims/data/vectordata	tabular feature data, structured by collection, region, time
/balticaims/data/licenses	licenses for all offered datasets
/balticaims/inputs	temporal location for input data to be inserted into xcube or geodb
/balticaims/cube	temporal location for generated cubes before they are moved to the S3 bucket
/ba	mount point for NFS gateway of balticaims S3 bucket
/opt/tlhj	JupyterHub software installation
/var/lib/postgresql/10/main	Postgresql database with geodb data and Keycloak config
/etc/apache2	Apache2 virtualhost configuration

Path	Content
/home/eouser/BalticAIMS	github repository with xcube configuration, xcube gen scripts and configurations and readmes, copies of Jupyter notebooks of baadmin and bauser
/home/eouser	links to xcube configurations, log files of services, temporal log files of xcube gen runs, credentials for S3 bucket access
/home/jupyter-baadmin	Jupyter notebooks and .env file of baadmin
/home/jupyter-bauser	Jupyter notebooks and .env file of bauser

Table 3-3: S3 bucket cloud storage directory structure for data

Path	Content
s3://balticaims/inputs	attic of input datasets for cube generation or geodb insertion, sorted by origin and collection, not operationally used
s3://balticaims/cubes	runtime cube zarr directory structures, one per data cube, currently 22 cubes for regions fi, go, mv, and complete Baltic Sea

The cubes are consolidated and optimised (empty chunks stripped), and most of them have an image pyramid added.

3.4 Domain configuration balticaims.eu

The project has reserved the domain name **balticaims.eu** for 2 years, starting from 14.2.2022. An additional domain park service was acquired from service provider "hostingpalvelu.fi". Host names for different BalticAIMS services are mapped to the ip address 185.178.84.146 of *balticcube* VM on Creodias using DNS A records.

Mapped hostnames include

- www.balticaims.eu
- balticaims.eu
- viewer.balticaims.eu
- hub.balticaims.eu
- xcube.balticaims.eu
- geodb.balticaims.eu
- auth.balticaims.eu
- data.balticaims.eu
- geoserver.balticaims.eu

Figure 3-3: DNS A Record configuration

This configuration causes URLs containing these hosts to be forwarded to the reverse proxy of BalticAIMS. Its configuration dispatches to the different BalticAIMS services as described in section 4.8 below.

3.5 Github configuration

For sharing the service configurations and data ingestion tools, a GitHub repository has been created. The repository contains

- Instructions and code recipes for creating the required data cubes
- Configuration and setup instructions for the BalticaIMS xcube services
- Jupyter notebooks for inserting and using geoDB datasets

Currently this repository is accessible for project partners only and hosted under Finnish Environment Institute's GitHub organization account. Repository can be opened for the public later.

The repository address is

https://github.com/sykefi/BalticAIMS

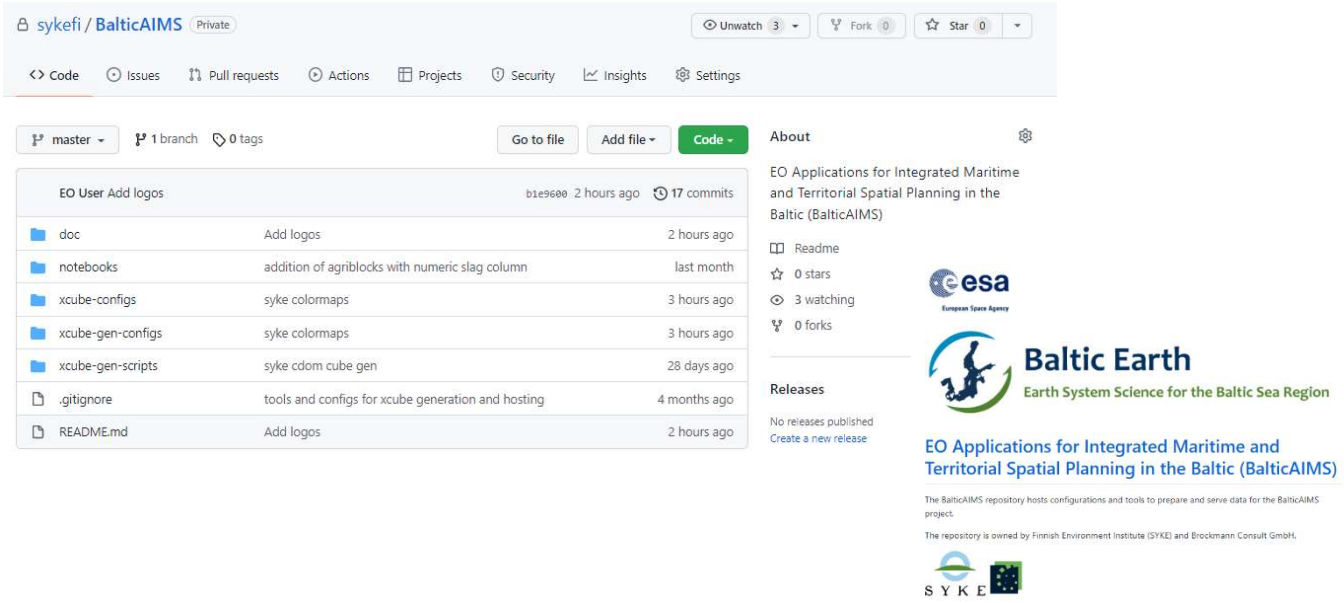

Figure 3-4: BalticAIMS Github Repository

4 Services setup

This section describes how the services for xcube, geodb, JupyterHub, and TARKKA are set up and configured for BalticAIMS.

4.1 xcube server

xcube and geodb software are installed into a miniconda environment. xcube is either installed from source available on github and installed into the conda environment, or it is installed as a conda package and patched afterwards.

```
sudo sysctl -w fs.inotify.max_user_watches=100000

sudo mkdir /balticaims
sudo chown eouser /balticaims
mkdir /balticaims/packagescd /balticaims/packages
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -b -p /balticaims/software/miniconda3-ba
eval "$(/balticaims/software/miniconda3-ba/bin/conda shell.bash hook)"
conda install -y -c conda-forge mamba

conda create --name xcube
conda activate xcube
mamba install -c conda-forge xcube rioxarray xcube_geodb jupyterhub postgis
cd /balticaims/software
git clone https://github.com/dcs4cop/xcube.git
cd xcube
mamba env create
python setup.py install

cd /balticaims/software
cd xcube-gen-balticaims
python setup.py install

# update xcube to support …
# update xcube_geodb to support openid_connect
#cd /balticaims/software
#patch xcube and geodb software
#git clone -b boe_xxx_for_balticaims https://github.com/dcs4cop/xcube-geodb.git
#diff xcube-geodb/xcube_geodb/core/geodb.py /balticaims/software/miniconda3-
ba/envs/xcube/lib/python3.9/site-packages/xcube_geodb/core/geodb.py
# cp xcube-geodb/xcube_geodb/core/geodb.py /balticaims/software/miniconda3-
ba/envs/xcube/lib/python3.9/site-packages/xcube_geodb/core/geodb.py
#diff xcube/xcube/core/gen/gen.py /balticaims/software/miniconda3-
ba/envs/xcube/lib/python3.9/site-packages/xcube/core/gen/gen.py
# cp xcube/xcube/core/gen/gen.py /balticaims/software/miniconda3-
ba/envs/xcube/lib/python3.9/site-packages/xcube/core/gen/gen.py
```

Figure 4-1: Instructions to set up conda environment and xcube

The configuration of xcube server lists the published data cubes with their data location in the balticaims S3 bucket and with properties for visualisation.

```
Datasets:
  - Identifier: FI_SYKE_TUR_1D
    Title: Finland SYKE S2 turbidity 1D
    FileSystem: obs
    Path: "balticaims/cubes/fi-syke-EO_HR_WQ_S2_TURB-1d.zarr"
```

```
    Endpoint: "https://cf2.cloudferro.com:8080"
    Region: "RegionOne"
    Style: syke_legends
  - Identifier: FI_HROC_L4D_1M
    Title: Finland HROC L4D monthly
    FileSystem: obs
    DatasetFormat: zarr
    Path: "balticaims/cubes/fi-hroc-l4d-monthly.zarr"
    Endpoint: "https://cf2.cloudferro.com:8080"
    Region: "RegionOne"
    Style: BAv4
…
Styles:
  - Identifier: BAv1
    ColorMappings:
      CHL:
        ColorBar: "jet"
        ValueRange: [.1, 20.]
      SPM:
        ColorBar: "gist_earth"
        ValueRange: [.0, 50]
      TUR:
        ColorBar: "gist_earth"
        ValueRange: [.0, 50]
      rgb:
          Red:
              Variable: B4
              ValueRange: [0.,0.3]
          Green:
              Variable: B3
              ValueRange: [0.,0.3]
          Blue:
              Variable: B2
              ValueRange: [0.,0.3]
…
# Metadata ------------
ServiceProvider:
  ProviderName: "BalticAIMS"
…
```

Figure 4-2: xcube server configuration ba-server-config-v1.yml with entries for cubes and styles

xcube server is started with

```
eval "$(/balticaims/software/miniconda3-ba/bin/conda shell.bash hook)"
conda activate xcube
export AWS_ACCESS_KEY_ID=… (insert EC2 credentials here)
export AWS_SECRET_ACCESS_KEY=… (insert EC2 credentials here)
nohup xcube --traceback serve -v --address 127.0.0.1 --port 8081 --prefix
http://xcube.balticaims.eu/api --revprefix http://xcube.balticaims.eu/api --config ~/ba-
server-config-v1.yml > ~/xcube.out &
tail -f ~/xcube.out
```

4.2 xcube viewer

The xcube viewer requires a node binary package. xcube viewer is installed from source available on github.

```
cd /balticaims/packages
wget ...node-v14.17.0-linux-x64.tar.xz
cd /balticaims/software
tar xf /balticaims/packages//node-v14.17.0-linux-x64.tar.xz
export PATH=/balticaims/software/node-v14.17.0-linux-x64/bin:$PATH

cd /balticaims/software
```

```
tar xf /balticaims/packages/node-v14.17.0-linux-x64.tar.xz
export PATH=/balticaims/software/node-v14.17.0-linux-x64/bin:$PATH
git clone https://github.com/dcs4cop/xcube-viewer.git
cd xcube-viewer
npm install -g yarn
yarn install
yarn build
```

Figure 4-3: Instructions to set up the xcube viewer software

The viewer gets its configuration by a rewrite rule in Apache. The rule points viewer.balticaims.eu/config/ to the content made available by the file server at www.balticaims.eu/config/ which is serving the files in directory /balticaims/www/config/ . The configuration itself is some styles and icons and the URL of the xcube server to connect with initially.

```
    RewriteRule ^/config/(.*)$ https://www.balticaims.eu/config/$1 [R=permanent,L]
```

Figure 4-4: Apache rewrite rule to let xcube viewer find its configuration

xcube viewer is started with

```
export PATH=/balticaims/software/node-v14.17.0-linux-x64/bin:$PATH
cd /balticaims/software/xcube-viewer
nohup yarn start > ~/viewer.out &
tail -f ~/viewer.out
```

4.3 Postgresql database server

The backend service of geodb is Postgresql with PostGIS. Postgresql version 10 and PostGIS version 2.5 are installed.

```
sudo apt install postgresql-10 postgresql-10-postgis-2.5 postgresql-10-postgis-2.5-scripts
sudo emacs /etc/postgresql/10/main/postgresql.conf
--
autovacuum = on
track_counts = on
data_directory = '/var/lib/postgresql/10/main'
hba_file = '/etc/postgresql/10/main/pg_hba.conf'
ident_file = '/etc/postgresql/10/main/pg_ident.conf'
--
sudo emacs /etc/postgresql/10/main/pg_hba.conf
--
local   all             all                                     peer
host    all             all             127.0.0.1/32            md5
--

# startup
sudo service postgresql start
netstat -an | grep 5432
```

Figure 4-5: Instructions to install Postgresql

4.4 Keycloak authentication server

Keycloak is a web service for authentication. It uses Postgresql as database backend for user management.

```
cd /balticaims/packages
wget https://github.com/keycloak/keycloak/releases/download/16.1.0/keycloak-16.1.0.zip
```

```
wget https://jdbc.postgresql.org/download/postgresql-42.3.1.jar
cd /balticaims/software
unzip /balticaims/packages/keycloak-16.1.0.zip
cd keycloak-16.1.0

# install database binding

mkdir -p modules/system/layers/keycloak/org/postgresql/main
cp /balticaims/packages/postgresql-42.3.1.jar
modules/system/layers/keycloak/org/postgresql/main
emacs -nw modules/system/layers/keycloak/org/postgresql/main/module.xml
emacs -nw standalone/configuration/standalone.xml
```

Figure 4-6: Instructions to install Keycloak and to update the configuration

Parts of the two configuration files to be adapted are shown in Figure 4-7 and Figure 4-8.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="org.postgresql">
    <resources>
        <resource-root path="postgresql-42.3.1.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

Figure 4-7: Part of module.xml keycloak configuration with adaptations for BalticAIMS

```
    <datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true" statistics-enabled="${wildfly.datasources.statistics-
enabled:${wildfly.statistics-enabled:false}}">
        <connection-url>jdbc:postgresql://localhost:5432/keycloak</connection-url>
        <driver>postgresql</driver>
        <pool>
            <max-pool-size>20</max-pool-size>
        </pool>
        <security>
            <user-name>cloaky</user-name>
            <password>...</password>
        </security>
    </datasource>
    <driver name="postgresql" module="org.postgresql">
        <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    </driver>
```

Figure 4-8: Part of standalone.xml keycloak configuration with adaptations for BalticAIMS

After Apache2 server configuration, an entry in standalone.xml needs to be added for the external URL:

<property name="frontendUrl" value="http://auth.balticaims.eu/auth"/>

The database is created empty, with a new user owning it.

```
sudo su - postgres
psql
--
create database keycloak;
create role cloaky;
alter role cloaky password '...';
grant all on database keycloak to cloaky;
--
```

Figure 4-9: Instructions to set up database for Keycloak

Next, Keycloak needs to be started with local access to create an admin user.

```
nohup bin/standalone.sh -b 127.0.0.1 > ~/keycloak.out &

http://localhost:8080/auth
--
martin
…
--
ps -elf|grep keycloak
kill
```

Figure 4-10: Admin user creation in Keycloak

Finally, geodb and JupyterHub are configured as “client” applications.

```
nohup bin/standalone.sh -b 127.0.0.1 > ~/keycloak.out &
http://localhost:8080/auth/admin
--
realm ba
client geodb
default client scope roles
client mapper username

client jupyterhub with redirect uris http://localhost:8000/*
(later http://hub.balticaims.eu/*,https://hub.balticaims.eu/*)

user baadmin password … temporary off
user bauser password balticaims temporary off
--
```

Figure 4-11: Registration of clients geodb and JupyterHub in Keycloak

4.5 geodb

The geodb Python software is included in the Conda environment (section 4.1). It will be used in Jupyter notebooks. To set up geodb a specific schema is deployed into a database geodb in Postgresql. Since the official schema of geodb is not working, we have created a branch in geodb with the fixed schema. In addition, a PostgREST server is put in front of Postgresql that is used by the geodb API and that uses Keycloak for authentication. The steps for installation are the following.

```
cd /balticaims/software
git clone -b boe_xxx_for_balticaims https://github.com/dcs4cop/xcube_geodb.git

sudo su - postgres
psql
--
create database geodb;
--

# use modified geodb.sql from branch
psql -d geodb -f /balticaims/software/xcube-geodb/xcube_geodb/sql/geodb.sql
psql geodb
--
alter role authenticator login password '...';
select geodb_register_user('baadmin', '...');
select geodb_register_user('bauser', '...');
--
```

Figure 4-12: Instructions to install geodb software and geodb database

PostgREST is installed as follows. It requires a configuration shown in Figure 4-14.

```
cd /balticaims/packages
wget https://github.com/PostgREST/postgrest/releases/download/v7.0.1/postgrest-v7.0.1-linux-
x64-static.tar.xz
cd /balticaims/software
mkdir postgrest
cd postgrest
tar xf /balticaims/packages/postgrest-v7.0.1-linux-x64-static.tar.xz
# edit geodbrest.conf, use first key in certs as jwt secret
```

Figure 4-13: Instructions to install PostgREST

```
server.oirt = "3001"
server-host = "127.0.0.1"
db-uri = "postgres://authenticator:...@127.0.0.1:5432/geodb"
db-schema = "public"
db-anon-role = "web_anon"
jwt-secret =
"{\"kid\":\"...\",\"kty\":\"RSA\",\"alg\":\"RS256\",\"use\":\"sig\",\"n\":\"...\",\"e\":\"...\"}"
role-claim-key = ".preferred_username"
```

Figure 4-14: PostgREST configuration geodbrest.conf

- The db-uri authenticator password … needs to be inserted from what has been set above.
- The jwt secret can be retrieved from Keycloak with
  `curl http://localhost:8080/auth/realms/ba/protocol/openid-connect/certs`

PostgREST is started with

```
nohup postgrest geodbrest.conf > ~/postgrest.out &
```

4.6 JupyterHub

The JupyterHub Python software is partially included in the Conda environment (section 4.1). The web server part is installed with a few steps (Figure 4-15) and needs configuration for authentication (Figure 4-16).

```
sudo su -
eval "$(/balticaims/software/miniconda3-ba/bin/conda shell.bash hook)"
conda activate xcube
cd /opt
curl -L https://tljh.jupyter.org/bootstrap.py | sudo -E python - --admin martin

# enable syslog for error analysis
apt list -a rsyslog
apt install rsyslog
```

Figure 4-15: Installation of JupyterHub web server software

- JupyterHub needs to be installed as root.
- The conda env is sourced to make available the right Python kernel to notebooks.
- An admin user is created.
- syslog shall be enabled to be able to see error messages.

```
users:
  admin:
  - martin
http:
  port: 8000
nobase_url: /jupyterhub/

auth:
  type: oauthenticator.generic.GenericOAuthenticator
  GenericOAuthenticator:
    client_id: jupyterhub
    oauth_callback_url: 'http://localhost:8000/oauth_callback'
```

```
    authorize_url: 'http://localhost:8080/auth/realms/ba/protocol/openid-connect/auth'
    token_url: 'http://localhost:8080/auth/realms/ba/protocol/openid-connect/token'
    userdata_url: 'http://localhost:8080/auth/realms/ba/protocol/openid-connect/userinfo'
    login_service: keycloak
    username_key: preferred_username
    userdata_params:
      state: state
  JupyterHub:
    authenticator_class: generic-oauth
```

Figure 4-16: Initial authentication configuration for JupyterHub in /opt/tljh/config/config.yaml

- The configuration names the admin user, the port, and different authentication URLs

There is a step to create the Jupyter accounts for baadmin and bauser. They need to be added as users to Keycloak. Then, maybe the first login creates their Linux accounts jupyter-baadmin and jupyter-bauser. Provisionally, the baadmin (and in the same way the bauser) account needs a configuration to access geodb. This is shown in Figure 4-17.

```
GEODB_API_SERVER_URL="https://geodb.balticaims.eu"
GEODB_API_SERVER_PORT=443
GEODB_AUTH_MODE="openid"
GEODB_AUTH_DOMAIN="https://auth.balticaims.eu/auth/realms/ba/protocol"
GEODB_AUTH_CLIENT_ID="geodb"
GEODB_AUTH_USERNAME="baadmin"
GEODB_AUTH_PASSWORD="..."
```

Figure 4-17: User configuration of access credentials to geodb in /home/jupyter-baadmin/.env for Jupyter notebooks

Notebooks that shall be visible to these two users are copied into the root directory of their accounts. baadmin gets the “insert” notebooks. bauser gets the “use” notebooks. The originals of the notebooks are maintained on GitHub in the BalticAIMS repo.

After Apache2 server configuration, /opt/tljh/config/config.yaml needs to be updated for the external URLs.

```
auth:
  type: oauthenticator.generic.GenericOAuthenticator
  GenericOAuthenticator:
    client_id: jupyterhub
    oauth_callback_url: 'http://hub.balticaims.eu/oauth_callback'
    authorize_url: 'http://auth.balticaims.eu/auth/realms/ba/protocol/openid-connect/auth'
    token_url: 'http://auth.balticaims.eu/auth/realms/ba/protocol/openid-connect/token'
    userdata_url: 'http://auth.balticaims.eu/auth/realms/ba/protocol/openid-connect/userinfo'
    login_service: keycloak
    username_key: preferred_username
    userdata_params:
      state: state
  JupyterHub:
    authenticator_class: generic-oauth
```

Figure 4-18: Updated JupyterHub configuration for access control once Apache2 reverse proxy is in place

4.7 Geoserver WFS

Geoserver implements WFS and WMS for the feature data. The geodb content is configured as a data source for this purpose. Geoserver directly accesses the Postgresql database backend, not the Python API of geodb.

After giving the admin user of Geoserver a new password, we create a workspace baadmin (Figure 4-19) and a store data source baadmin (Figure 4-20).

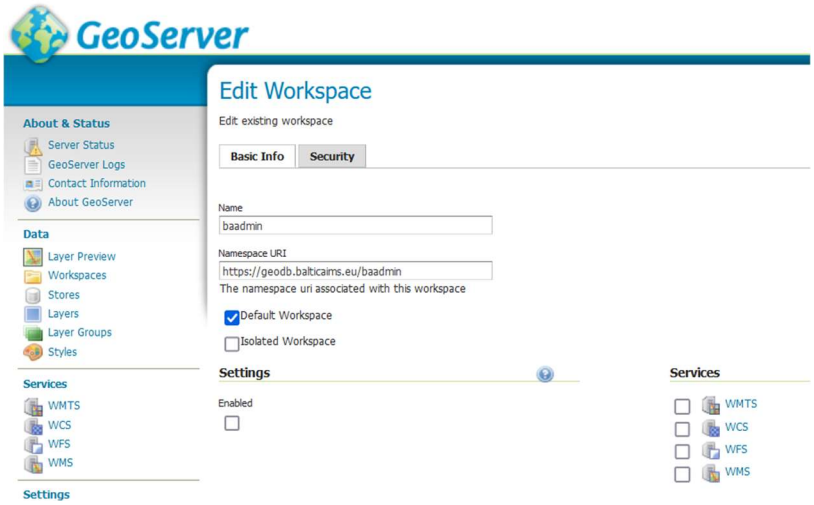

Figure 4-19: Geoserver configuration of workspace baadmin

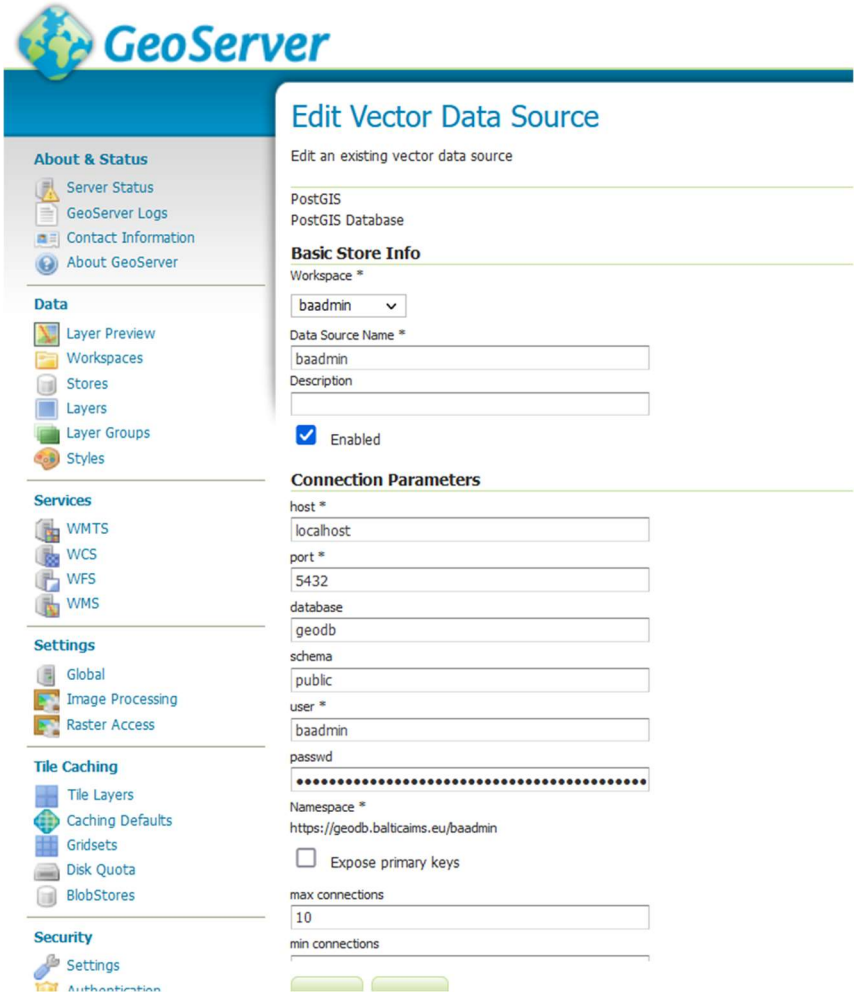

Figure 4-20: Geoserver configuration of store data source baadmin with access to baadmin's database geodb

There is a Geoserver API to publish feature datasets available in the vector data store. The API can be used in a Jupyter notebook after inserting the data into geodb. To make this API accessible from a notebook we add 3 lines to /home/jupyter-baadmin/.env:

```
GEOSERVER_URL="http://localhost:8082/geoserver"
GEOSERVER_USER="baadmin"
GEOSERVER_PASSWORD="…"
```

Figure 4-21: Configuration of Geoserver access for publication of collections by user baadmin in Jupyter notebooks

With this configuration, baadmin can publish new feature collections in Geoserver.

4.8 Apache server

The apache2 server is the reverse proxy of the different services of BalticAIMS. It provides the encrypted HTTPS connections and delegates requests with different virtual hosts to the respective service processes. It is configured for the virtual hosts that all share a common IP address.

Apache2 is installed and the data directories and configurations are as follows.

```
sudo apt update
sudo apt install apache2
mkdir /balticaims/data
mkdir /balticaims/www
cd /etc/apache2
ls sites-available
```

Figure 4-22: Apache2 installation with location of data directories and virtual host configurations

One virtual host configuration is done in three files, one for HTTP, one for HTTPS, and one with the common configuration. Figure 4-23, Figure 4-24, and Figure 4-25 show this for the geo file server data.balticaims.eu.

```
<VirtualHost *:80>
    Include sites-available/data.common
</VirtualHost>
```

Figure 4-23: HTTP configuration data.conf for the geo data server data.balticaims.eu

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    Include sites-available/data.common
    SSLCertificateFile /etc/letsencrypt/live/balticaims.eu/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/balticaims.eu/privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```

Figure 4-24: HTTPS configuration data-le-ssl.conf for the geo data server data.balticaims.eu

- The generation of the certificates is described at the end of this subsection.

```
    ServerAdmin webmaster@localhost
    ServerName data.balticaims.eu
    ServerAlias data

    ErrorLog ${APACHE_LOG_DIR}/data-error.log
    CustomLog ${APACHE_LOG_DIR}/data-access.log combined

    DocumentRoot /balticaims/data
    <Directory /balticaims/data>
        Options +Indexes
        Require all granted
    </Directory>
```

Figure 4-25: Common configuration data.common for the geo data server data.balticaims.eu

data.common

- defines the server name data.balticaims.eu

- defines names for the log files
- defines which directory tree to serve, /balticaims/data

www.common is similar to data.common

- with an additional line `Header set Access-Control-Allow-Origin "*"`

xcube.common is a service available at port 8081 on localhost. Its configuration is shown in Figure 4-26.

```
    ServerAdmin webmaster@localhost
    ServerName xcube.balticaims.eu
    ServerAlias xcube

    ErrorLog ${APACHE_LOG_DIR}/xcube-error.log
    CustomLog ${APACHE_LOG_DIR}/xcube-access.log combined

    RewriteEngine On
    ProxyPreserveHost On
    RewriteCond %{HTTPS} on
    RewriteRule ^$ https://%{HTTP_HOST}/ [R=permanent,L]
    RewriteRule ^$ http://%{HTTP_HOST}/ [R=permanent,L]
    RewriteRule ^/(.*)$ http://127.0.0.1:8081/$1 [P]

    Header set Access-Control-Allow-Origin "*"
```

Figure 4-26: Common configuration xcube.common for the xcube server xcube.balticaims.eu

xcube.common

- uses rewrite rules to map external URLs to local ones with port 8081

viewer.common is similar,

- without the ProxyPreserveHost line
- with port 3002 instead of 8081
- with an additional line to map accesses to the xcube viewer configuration to an URL of www.balticaims.eu
  RewriteRule ^/config/(.*)$ https://www.balticaims.eu/config/$1 [R=permanent,L]

geodb.common is similar,

- without the ProxyPreserveHost line
- with port 3001 instead of 8081
- without the Header set Accces-Control-Allow-Origin “*” line

geoserver.common is similar,

- with port 8082 instead of 8081

hub.common contains additional rules for web sockets. It is shown in Figure 4-27.

```
    ServerAdmin webmaster@localhost
    ServerName hub.balticaims.eu
    ServerAlias hub

    ErrorLog ${APACHE_LOG_DIR}/hub-error.log
    CustomLog ${APACHE_LOG_DIR}/hub-access.log combined

    RewriteEngine On
    ProxyPreserveHost On

    RewriteCond %{HTTPS} on
    RewriteRule ^$ https://%{HTTP_HOST}/ [R=permanent,L]
    RewriteRule ^$ http://%{HTTP_HOST}/ [R=permanent,L]

    RewriteCond %{HTTP:Connection} Upgrade [NC]
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteRule /(.*) ws://127.0.0.1:8000/$1 [P,L]
    RewriteRule /(.*) http://127.0.0.1:8000/$1 [P]
```

Figure 4-27: Common configuration hub.common for the JupyterHub server hub.balticaims.eu

The Apache2 server needs SSL certificates to serve via HTTPS. They can be generated with letsencrypt as shown in Figure 4-28.

```
apt-get update
apt-get install certbot python3-certbot-apache
certbot --apache
# enter email address (used for urgent renewal and security notices) (Enter 'c' to cancel):
bc-it@brockmann-consult.de
(A)gree/(C)ancel: A
(Y)es/(N)o: N
blank to select all options shown (Enter 'c' to cancel): c

(cancel because I don't like that it will randomly pick one instead of using what I want as
the main CN...also what I wanted wasn't listed (without www))

# make sure the default domain is defined
root@balticcube:/etc/apache2/sites-available # vim www.common
 ServerName balticaims.eu

# make sure the default site is the first in sort order
root@balticcube:/etc/apache2/sites-enabled # mv www.conf 000-www.conf
root@balticcube:/etc/apache2/sites-enabled # service apache2 restart
certbot --apache -n -d balticaims.eu
certbot --apache -n --expand -d
balticaims.eu,www.balticaims.eu,data.balticaims.eu,viewer.balticaims.eu,xcube.balticaims.eu,ge
odb.balticaims.eu,auth.balticaims.eu,hub.balticaims.eu
```

Figure 4-28: Creation of SSL certificates for the Apache2 server

There is a very simple entry page with just links to the BalticAIMS services (Figure 4-29). It is served at https://www.balticaims.eu .

```
<html>
<head><title>BalticAIMS</title></head>
<body>Welcome to the BalticAIMS infrastructure.
<p><a href="https://data.balticaims.eu/">File server</a></p>
<p><a href="https://hub.balticaims.eu/">Jupyter Hub</a></p>
<p><a href="https://viewer.balticaims.eu/">xcube viewer</a></p>
<p><a href="https://xcube.balticaims.eu/">xcube REST server</a></p>
<p><a href="https://geodb.balticaims.eu/">geoDB REST server</a></p>
<p><a href="https://auth.balticaims.eu/">Keycloak auth server</a></p>
<p><a href="https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml">xcube WMTS</a></p>
<p><a href="https://geoserver.balticaims.eu/geoserver/wfs?request=GetCapabilities">geoDB
WFS</a></p>
<p><a href="https://geoserver.balticaims.eu/geoserver/wms?request=GetCapabilities">geoDB
WMS</a></p>
</body>
</html>
```

Figure 4-29: index.html with links to the BalticAIMS services

4.9 TARKKA instance

The TARKKA front-end is built and deployed using the node package manager (*npm*). The target server is defined in the *deplConf.json* configuration file with the following contents:

```
{
      "prod" : {
                "buildCommand": "npm run build",
                "paths": [<list of folderpaths into which the frontend is published>]
     }
}
```

The actual command to publish the frontend is "npm run publish:prod".

To access spatial data for map visualizations, OGC-compliant WMS or WMTS services are used to provide well-performing data access. Xcube WMTS layers are configured in TARKKA by adding entries to *layerCatalog.ts* – configuration item.

```
const layerCatalog: LayerCatalogItem[] = [

  {
    type: LayerType.XCUBE_WMTS,
    url: 'https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml',
    nameTranslationKey: 'layers.layerNames.balticaims_xcube_turb',
    layer: 'FI_SYKE_TUR_1D.turb',
    attribution: 'BalticAIMS, SYKE',
    id: 'balticaims_xcube_turb_s2',
    matrixSet: 'TileGrid_0',
    opacity: 1.0,
    zIndex: 100,
    defaultLayer: false,
    isConfidential: false,
    temporalType: LayerTemporalType.XCUBE_WMTS,
    featureinfoType: LayerFeatureinfoType.OMIT,
    colormap: {
      name: 'ba-syke-turbidity-legend.cpd',
      stops: [0, 60],
    },

  },
  {
    type: LayerType.XCUBE_WMTS,
    url: 'https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml',
    nameTranslationKey: 'layers.layerNames.balticaims_xcube_turb_hroc',
    layer: 'FI_HROC_L4D_1M.TUR_mean',
    attribution: 'Copernicus',
    id: 'balticaims_xcube_turb_hroc',
    matrixSet: 'TileGrid_0',
    opacity: 1.0,
    zIndex: 101,
    defaultLayer: false,
    isConfidential: false,
    temporalType: LayerTemporalType.XCUBE_WMTS,
    featureinfoType: LayerFeatureinfoType.OMIT,
    colormap: {
      name: 'ba-syke-turbidity-legend.cpd',
      stops: [0, 60],
    },
          ...
```

To access timeseries data, TARKKA can utilize OData-standard compliant APIs, or any data providers providing the timeseries data in JSON/GeoJSON format Interfaces for time-series extraction are configured in *timeseriesCatalog.ts*

```
const timeseriesCatalog: TTimeseriesDefinition[] = [
  {
    id: 'balticaims_xcube_turb_s2_1',
    name: 'timeseries.definitions.xcube-turb',
    parameter: PhysicalParameter.TURBIDITY,
    unit: PhysicalUnit.FNU,
    providertype: TimeseriesProviderType.XCUBE,
    timeseriestype: TimeseriesType.BASICSTATS,
    url: 'https://xcube.balticaims.eu',
    datasetId: 'FI_SYKE_TUR_1D',
    variableId: 'turb',
    availableTimeWindow: ['2018-04-01', '2021-08-01'],
  },
  {
    id: 'balticaims_xcube_turb_hroc_1',
    name: 'timeseries.definitions.xcube-turb-hroc',
    parameter: PhysicalParameter.TURBIDITY,
    unit: PhysicalUnit.FNU,
    providertype: TimeseriesProviderType.XCUBE,
    timeseriestype: TimeseriesType.BASICSTATS,
    url: 'https://xcube.balticaims.eu',
    datasetId: 'FI_HROC_L4D_1M',
    variableId: 'TUR_mean',
    availableTimeWindow: ['2018-04-01', '2021-08-01'],
  } ,
  ...
```

Finally, the BalticAIMS use cases are defined as new *dataCollections* in TARKKA configuration. *dataCollections* consist of items in the *layerCatalog* and *timeseriesCatalog, f*or example: *BalticAIMS_datacollection.ts*

```
// This is the data collection available for the TARKKA instance.
export const BalticAimsDatasetCollection: TDatasetCollection = {
  id: 'tarkka-balticaims-datasets-usecase-X',
  name: 'dataset_collection.balticaims.name', //'BalticAIMS datasets (use case Dredging)',
  description:
    'Contains datasets for demonstrating the dredging incidence use case.',
  tags: ['balticaims', 'dredging'],
  contents: [
    {
      id: 'turbidity',
      name: 'dataset_collection.balticaims.item_name.turbidity',
      icon: 'icon_FaGlobeEurope',
      description: 'Coastal water turbidity estimate from Sentinel-2 MSI',
      datasets: [
        {
          id: 'balticaims-turbidity-syke',
          name: 'dataset_collection.balticaims.item_name.turbidity_syke',
          datasourceclass: DatasourceClass.EO,
          instrument: EarthObservationInstrument.S2_MSI,
          parameter: PhysicalParameter.TURBIDITY,
          description: 'Water turbidity (Sentinel-2 MSI)', // add this to i18n-translations
          attribution: 'attribution.balticaims.turbidity_syke',
          mapDefinitionIds: ['balticaims_xcube_turb_s2'],
```

```
          timeseriesDefinitionIds: ['balticaims_xcube_turb_s2_1'],
          availableTimewindow: ['2020-06-01', '2020-08-01'],
        },
        {
          id: 'balticaims-turbidity-hiroc',
          name: 'dataset_collection.balticaims.item_name.turbidity_hroc',
          datasourceclass: DatasourceClass.EO,
          instrument: EarthObservationInstrument.S2_MSI,
          parameter: PhysicalParameter.TURBIDITY,
          description: 'Water turbidity (HROC)', // add this to i18n-translations
          attribution: 'attribution.balticaims.turbidity_hroc',
          mapDefinitionIds: ['balticaims_xcube_turb_hroc'],
          timeseriesDefinitionIds: ['balticaims_xcube_turb_hroc_1'],
          availableTimewindow: ['2020-06-01', '2020-08-01'],
        },
        {
          id: 'vesla-turbidity-syke',
          name: 'dataset_collection.balticaims.item_name.turbidity_vesla',
          datasourceclass: DatasourceClass.MS,
          parameter: PhysicalParameter.TURBIDITY,
          description: 'Water turbidity (VESLA)', // add this to i18n-translations
          attribution: 'attribution.balticaims.turbidity_syke',
          mapDefinitionIds: ['reference_station_points'],
          timeseriesDefinitionIds: ['vesla_turbidity'],
          availableTimewindow: ['2000-01-01', '2022-01-01'],
        },
      ],
    },
    {
      id: 'balticaims-gisdata',
      name: 'dataset_collection.balticaims.item_name.gis_data',
      icon: 'icon_BsLayersHalf',
      description: 'Dredging areas off-shore Helsinki',
      datasets: [
        {
          id: 'balticaims-dredgingareas',
          name: 'dataset_collection.balticaims.item_name.dredging_areas',
          datasourceclass: DatasourceClass.GIS,
          description: 'Dredging areas off-shore Helsinki', // add this to i18n-translations
          attribution: 'attribution.balticaims.dredginareas',
          mapDefinitionIds: ['balticaims_dredging_hki'],
        },
        {
          id: 'ref_regions',
          name: 'dataset_collection.gis.item_name.ref',
          datasourceclass: DatasourceClass.GIS,
          description: 'Reference station regions', // add this to i18n-translations
          attribution: 'attribution.ref_regions',
          mapDefinitionIds: ['reference_stations_areas_noHoles'],
        },
        {
          id: 'nature-reserves-fi',
          name: 'layers.layerNames.nature_reserves',
          datasourceclass: DatasourceClass.GIS,
          description: 'Nature reserver, Finland', // add this to i18n-translations
          attribution: 'attribution.balticaims.dredginareas',
          mapDefinitionIds: ['nature_reserves'],
        },
      ],
    },
    {
      id: 'rgb',
```

```
      name: 'dataset_collection.rgb.name',
      icon: 'icon_BsSliders',
      description: 'Sentinel-2 true color data',
      datasets: [
        {
          id: 's2_rgb',
          name: 'dataset_collection.rgb.item_name.s2',
          datasourceclass: DatasourceClass.EO,
          description: 'Sentinel-2 true color data', // add this to i18n-translations
          attribution: 'attribution.copernicus.sentinel2_rgb',
          mapDefinitionIds: ['sentinelhub_rgb_s2'],
        },
        {
          id: 's3_olci_rgb',
          name: 'dataset_collection.rgb.item_name.s3',
          datasourceclass: DatasourceClass.EO,
          description: 'Sentinel-3 Olci true color data', // add this to i18n-translations
          attribution: 'attribution.copernicus.sentinel3_rgb',
          mapDefinitionIds: ['sentinelhub_rgb_s3'],
        },
      ],
    },
  ],
}
```

5 References

Doc ID	Title	Origin, Version, Date
[Portfolio]	D2.1 BalticAIMS service portfolio definition	ESA project report, v1.1, 05.10.2021
[Platform]	D2.2 BalticAIMS data and platform provisioning plan	ESA project report, v1.0, 16.09.2021
[ChainSpec]	D2.3 BalticAIMS service delivery chain specification	ESA project report, v1.0, 16.09.2021
[SVR]	D3.1 BalticAIMS service chain verification report	ESA project report, v1.0, 24.03.2022