



**ODYSSEA**

Operating a network of integrated observatory  
systems in the Mediterranean Sea

## Project Deliverable Report

Deliverable Number: 6.4

Deliverable Title: Platform Operational, Maintenance and user Manual

Author(s): Nicolas Granier

Work Package Number: 6



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 727277

ODYSSEA Project Information	
Project full title	ODYSSEA: Operating a network of integrated observatory systems in the Mediterranean Sea
Project acronym	ODYSSEA
Grant agreement number	727277
Project coordinator	Georgios Sylaios, DUTH
Project start date and duration	1 <sup>st</sup> June 2017, 54 months
Project website	<a href="http://odysseaplatform.eu/">http://odysseaplatform.eu/</a>

Deliverable Information	
Work package number	6
Work package title	Platform Development, Operation and Maintenance
Deliverable number	6.4
Deliverable title	Platform Operational, Maintenance and user Manual
Description	Platform Operational, Maintenance and User Manual
Lead beneficiary	DUTH
Lead Author(s)	CLS Nicolas Granier / Salima El-Mokhtari
Contributor(s)	
Revision number	1.0
Revision Date	24-11-2021
Status (Final (F), Draft (D), Revised Draft (RV))	F

Dissemination level (Public (PU), Restricted to other program participants (PP), Restricted to a group specified by the consortium (RE), Confidential for consortium members only (CO))	PU
---	----

Document History			
Revision	Date	Modification	Author
0.1	12/11/2021	Initial draft	Salima El-Mokhtari / Nicolas Granier
1.0	24/11/2021	Final draft	Georgios Sylaios

Approvals				
	Name	Organisation	Date	Signature (initials)
Coordinator	Georgios Sylaios	DUTH	24/11/2021	GS
WP Leaders	Nicolas Granier	CLS	12/11/2021	NG

#### **PROPRIETARY RIGHTS STATEMENT**

This document contains information, which is proprietary to the ODYSSEA consortium. Neither this document, or the information contained within may be duplicated, used, or communicated except with the prior written permission of the ODYSSEA coordinator.

## Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>9</b>
1.1	Operation .....	9
1.2	Architecture .....	10
<b>2</b>	<b>K8S MARINOMICA DEPLOYMENT .....</b>	<b>11</b>
2.1	Project recovery .....	11
2.2	Installation from scratch .....	12
2.2.1	Front-End Installation .....	12
2.2.2	Installation of the Catalogue engine .....	14
2.2.3	Installing the DataCollection from scratch .....	15
2.2.4	NRPE installation from scratch .....	16
2.2.5	Product Factory: TRIX/WavePower installation from scratch .....	17
2.3	Update.....	18
2.3.1	Front-End Update .....	18
2.3.2	Catalogue engine update .....	19
2.3.3	Updating the DataCollection .....	20
2.4	Configuration .....	22
2.4.1	Configuration of GeoNetwork .....	22
2.4.2	Datacollection Job Configuration .....	23
2.4.3	MongoDB configuration .....	24
2.4.3.1	Managing user rights with MongoDB .....	24
2.4.3.2	Create a dump of the Mongo DB .....	25
2.4.3.3	Create a dump of the Mongo DB .....	25
2.4.4	Product Factory Configuration .....	26
2.4.4.1	Script execution of daily jobs .....	26
2.4.4.2	Script tracking logs .....	27
2.4.4.3	Script listing the jobs .....	27
2.4.4.4	Script listing the jobs .....	28

2.4.4.5	host, hardware settings.....	29
2.4.4.6	Motu host, software settings .....	29
2.4.4.7	External interfaces .....	30
2.4.4.8	Several Motu instances on a same host.....	30
2.4.5	Install Motu from scratch .....	30
2.4.5.1	Install Motu software, for example on a Dissemination Unit .....	31
2.4.5.2	Install Motu theme (public static files) .....	31
2.4.5.3	On a central server .....	32
2.4.5.3.1	Directly on Motu Apache tomcat server .....	32
2.4.5.4	Check installation .....	33
2.4.5.4.1	Start motu .....	33
2.4.5.4.2	Check messages on the server console .....	33
2.4.5.4.3	Check Motu web site available .....	33
2.4.5.4.4	Check Motu logs .....	33
2.4.6	Motu Configuration.....	33
2.4.6.1	Configuration directory structure .....	34
2.4.6.2	Business settings .....	34
2.4.6.2.1	motuConfiguration.xml: Motu business settings.....	34
2.4.6.2.2	Attributes defined in motuConfig node.....	35
2.4.6.3	System settings .....	43
2.4.6.3.1	motu.properties: Motu system settings .....	43
2.4.6.3.2	Java options .....	44
2.4.6.3.2.1	Tomcat network ports.....	44
2.4.6.3.2.2	CAS SSO server .....	45
2.4.6.4	Log settings .....	47
2.4.6.4.1	Motu queue server logs: motuQSlog.xml, motuQSlog.csv .....	47
2.4.6.5	Theme and Style.....	49
<b>2.5</b>	<b>Installation of the Product Factory .....</b>	<b>49</b>
2.5.1	Development Docker .....	49
2.5.1.1	Preparation of the environment .....	49
2.5.1.2	Launch of the dev service .....	51

2.5.2	Production Docker.....	51
2.5.2.1	Preparation of the environment .....	51
2.5.2.2	Launch of the prod.....	53
2.5.3	Performance of Nginx and uwsgi .....	53
2.5.4	Nginx and uwsgi log formatting .....	54
<b>3</b>	<b>OPERATOR INCIDENT PROCEDURE.....</b>	<b>55</b>
3.1	Nagios - Ingestion.....	55
3.2	Nagios - Export Folder .....	57
3.3	Nagios - FTP CMEMS.....	59
3.4	Nagios - HTTP check .....	62
3.5	Nagios - Check Pods.....	63
3.6	Nagios - Check PVC.....	64
3.7	Nagios - Check PVC Size .....	65
3.8	Nagios – Product Factory Trix / WavePower files .....	66
3.9	Nagios - Check Pod Restart .....	68
<b>4</b>	<b>IE INCIDENT PROCEDURE.....</b>	<b>70</b>
4.1	BDD H2 - Corrupted.....	70
4.2	Extension of a PVC.....	71
4.3	Aquasafe file not produced.....	73
4.4	Product Factory: TRIX files not produced .....	75
4.5	Product Factory: Wave Power file not produced .....	78
4.6	Blocked ingestion .....	80
4.7	Ingestion Gloss blocked .....	83
4.8	Restoration of the Postgis/ Postgres/ Geonetwork DB .....	85
4.8.1	Blocked ingestion .....	85
<b>5</b>	<b>APPENDIX.....</b>	<b>89</b>
5.1	Organization .....	89
5.1.1	Operations Engineer .....	89

5.1.2	Product Engineer.....	89
5.1.3	Developers .....	89
<b>5.2</b>	<b>K8S / Rancher controls .....</b>	<b>90</b>
5.2.1	Connection to the K8S PROD cluster.....	90
5.2.2	Update the Config file to access the K8SPROD1 cluster .....	91
5.2.2.1	Update of the config file to access the k8sprod1 cluster (Odyssea context) .....	91
5.2.2.2	Incorrect Odyssea context .....	92
5.2.3	Stop Restart & Useful Commands.....	92
5.2.4	Viewing log files .....	94
5.2.5	Introduction to Rancher .....	95
5.2.5.1	Connecting to Rancher .....	95
5.2.5.2	View PODS logs .....	96
5.2.5.3	Connecting to a POD (Bash window) .....	97
5.2.5.4	Follow an HTTP link from the application .....	98



## Table of Figures

<i>Figure 1.1 Block diagram for marinomica components, interrelations and data flux. ....</i>	<i>10</i>
<i>Figure 1.2 Marinomica Platform architecture.....</i>	<i>10</i>

## 1 Introduction

ODYSSEA is a web application that gives access to marine geographical and scientific data, metrics, and surveys and makes them available to end-users. It provides a large catalogue of meteorologic and oceanographic data related to the Mediterranean basin.

Through the Marinomica interface, the user will have access to services of:

- Visualization of sea beacons
- Layer display
- Download of data in different formats (csv, svg, netCDF, ...)

Access URL to the application: <https://marinomica.com/>

### 1.1 Operation

The Marinomica application includes several individual components (see below the functional diagram)

- Front-End, which has been implemented by the company BlueLobster IT, a module that manages the application interface and the display of the different data (mapped, in situ);
- DataCollection, which allows the recovery and formatting of data on the FTP CMEMS and by HTTP for Gloss (Global Sea Level Observing System), and thus allows the ingestion of data in the BDD. This component was created by the company Hidromod;
- Catalog-engine (created by Edisoft) which allows:
  - A recovery of files from the Data Collection component and an ingestion of the data in the SOS DataBase.
  - The harvesting of a CSW catalogue (datastore) and the CMEMS catalogue to allow the display of mapped data.
- Product Factory (developed by CLS) component that will generate all netcdf files containing data on the various in-house developed indicators, like the TRIX and the Wave Power products.

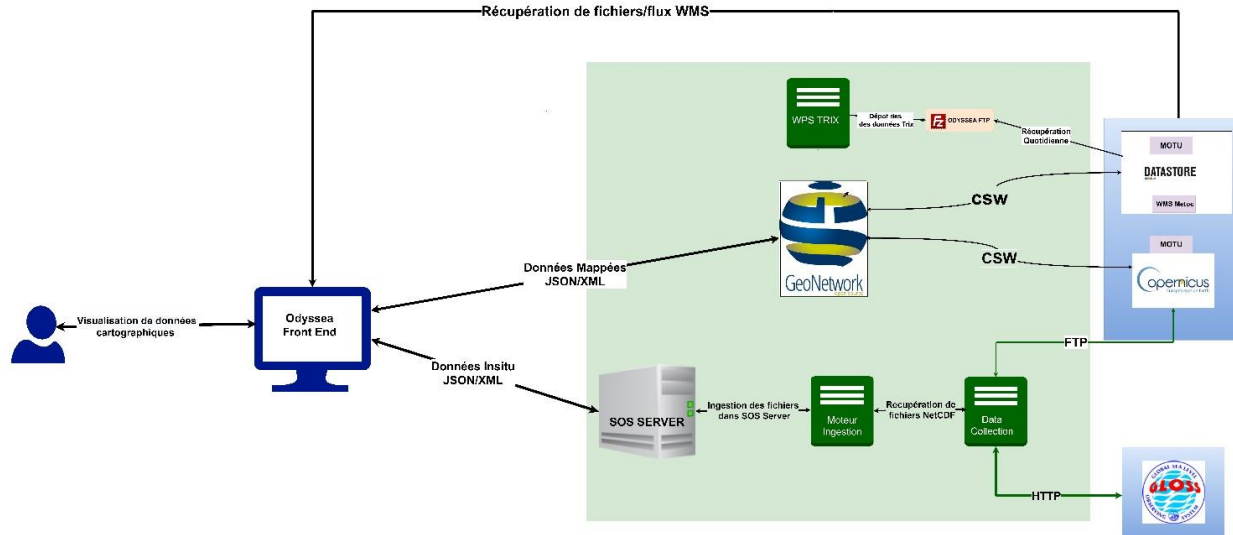


FIGURE 1.1 BLOCK DIAGRAM FOR MARINOMICA COMPONENTS, INTERRELATIONS AND DATA FLUX.

## 1.2 Architecture

The Marinomica application has been ported to K8S in January 2020. Below is a complete diagram of the current architecture:

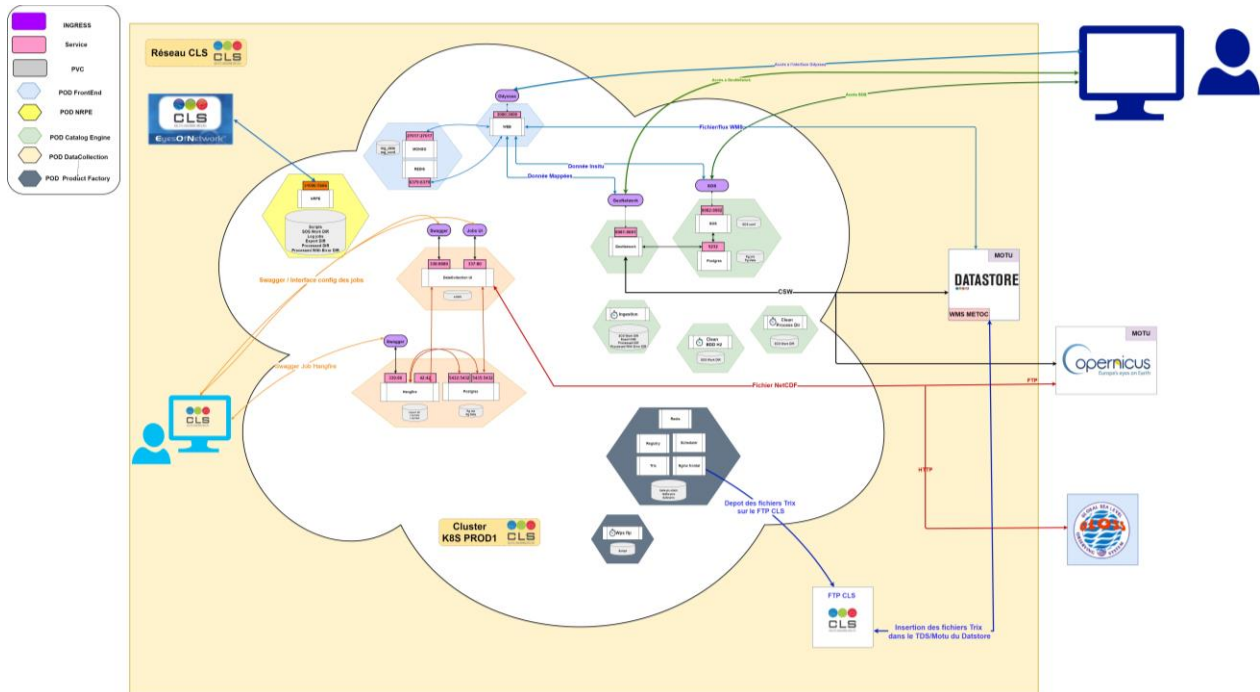


FIGURE 1.2 MARINOMICA PLATFORM ARCHITECTURE

## 2 K8S Marinomica Deployment

### 2.1 Project recovery

In order to deploy the Marinomica application you need to set up a deployment environment locally or on your VM cloud.

- Get the Odyssea project on Gitlab

```
root@xxxxxx:~# git clone https://gitshare.cls.fr/odyssea/odysseadeployment.git
```

- Install the kubectl client (<https://kubernetes.io/fr/docs/tasks/tools/install-kubectl/>)
- Download the latest release with the following command:

```
root@xxxxxx:~ curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
```

- Make the kubectl binary executable

```
root@xxxxxx:~ chmod +x ./kubectl
```

- Move the binary to your PATH

```
root@xxxxxx:~ sudo mv ./kubectl /usr/local/bin/kubectl
```

- Test to make sure the version you have installed is up to date:

```
root@xxxxxx:~ kubectl version
```

- To be able to access and modify the PROD/QO cluster you need to create a .kube directory which will contain the configuration files

```
root@xxxxxx:~ cd /home/ubuntu/  
root@xxxxxx:~ mkdir . kube
```

- Copy the QO and PROD configuration file to the . kube directory
- Your .kube folder should now contain these two configuration files

```
root@xxxxxx:~/odyssea/. kube# ls -lrt  
-rw-rw-rw- 1 ubuntu 2658 Jan 14 08:21 config.odyssea.prod  
-rw-rw-rw- 1 ubuntu 1852 Jan 27 08:19 config.odyssea.qo
```

- Edit your . bashrc with the path to these two files (an example line added to the . basrc below)

```
# K8S config
```

```
export KUBECONFIG=/home/ubuntu/odyssea/.kube/config.odyssea.qo:/home/ubuntu/odyssea/.kube/config.odyssea.prod
```

- Source your . bashrc

```
root@xxxxxxx:~# . . bashrc
```

- Set kubectl to the desired access context (QO or PROD)

```
root@xxxxxxx:~# kubectl config get-contexts

CURRENT NAME CLUSTER AUTHINFO NAMESPACE

k8s-fs-qt1 u-eqtu3gi2lg

* k8s-prod1

k8s-prod1-fqdn k8s-prod1

root@xxxxxxx:~# kubectl config use-context k8s-fs-qt1 (or kubectl config use-context k8s-prod1 for the prod)

Switched to context "k8s-fs-qt1".
```

- Perform a test to see if you have access to the cluster by retrieving for example the list of available pods

```
root@xxxxxxx:~# kubectl get pods -n odyssea-qo

NAME READY STATUS RESTARTS AGE

catalog-engine-7f689d678b-5lp9l 2/2 Running 0 6d17h

catalog-engine-clean-processed-dir-cronjob-1580086740-b6sr4 0/1 Completed 0 9h

catalog-engine-cronjob-1580119200-jt89s 0/1 Completed 0 10m

data-collection-7569556c87-87f4d 1/1 Running 0 44h

data-collection-hangfire-9dfb46d5f-r4vpf 2/2 Running 0 44h

frontend-odyssea-6958749564-jrpvj 1/1 Running 0 44h
```

## 2.2 Installation from scratch

### 2.2.1 Front-End Installation

- Connect to the deployment server
- Go to the deployment file

```
root@xxxxxxx: cd /XXXXXXXX/odyssea-deployment/k8s/cls/odyssea/
```

- The deployment of the Marinomica frontend is done in two steps: We will deploy the backend (redis and mongo), then the front end of the system.
- Run the following command with the parameters corresponding to your case

```
root@xxxxx: ./deploy-frontend-back-k8s-odyssea.sh < typecluster>
```

Cluster	Value for the typecluster parameter
QO	odysseaqo
Production	odyssea

- Below are the answers to the questions asked

```
Do you want to create pvc (y/n)? y
Do you want to create pods (dp files) (y/n)? y
Do you want to create services (y/n)? y
Do you want to create cronjob (y/n)? y
Do you want to copy script (y/n)? y
```

- View the status of the deployed pods and verify that the frontend-back-XXXX pod has the status "Ready: True

```
root@xxxxxx:~/odyssea/odyssea-deployment# kubectl get pods -n < typecluster>
root@xxxxxx: kubectl describe pod frontend-back-XXXXXXXXXX -n < typecluster>
```

- Connect to the front-end pod (mongo), create the admin user, and launch the restore from the present dump

```
root@xxxxxxx:~/odyssea/odyssea-deployment# kubectl exec -it frontend-back-XXXXXXX -c
mongo bash -n < typecluster>
root@frontend-back-5f6c7cbcd4-mx6g7:/db: mongo admin --username admin --password
UDW6cDyS7hm2xHrF < script.js
root@frontend-back-5f6c7cbcd4-mx6g7:/db: mongorestore --username admin --password
"UDW6cDyS7hm2xHrF" -d odysseaplatform "mongodump-2020-01-07/odysseaplatform/"
```

- Check that the tables are created correctly

```
root@frontend-odyssea-5f6c7cbcd4-mx6g7:/db:mongo -u admin -p UDW6cDyS7hm2xHrF --
authenticationDatabase odysseaplatform
>use odysseaplatform;
>show collections;
```

- We will proceed to the installation of the frontend to execute the following command with the parameters corresponding to your case (odysseaqa or odysseaprod)

```
root@xxxxxx: ./deploy-k8s-odyssea.sh frontend < typecluster>
```

- Below are the answers to the questions asked

```
Do you want to create pvc (y/n)?y
Do you want to create config map (y/n)?y
Do you want to create pods (dp files) (y/n)?y
Do you want to create services (y/n)?y
Do you want to create ingress (y/n)?y
Do you want to create cronjob (y/n)?y
```

- Check that the frontend interface is launched by testing the links created with the ingress. The link is available via the command below ( for the frontend the url is frontend.odyssea-dev.qt.cls.fr )

```
root@xxxxxx:~/odyssea/odyssea-deployment# kubectl get ingress -n < typecluster>

data-collection                               data-collection.odysseaqa.10.99.0.100.xip.io
10.99.0.100,10.99.0.55,10.99.0.62

frontend frontend.odyssea-dev.qt.cls.en 80 46s

geonetwork geonetwork.odyssea-dev.qt.cls.fr 10.99.0.100,10.99.0.55,10.99.0.62 80

hangfire hangfire.odysseaqa.10.99.0.100.xip.io 10.99.0.100,10.99.0.55,10.99.0.62 80

sos-server sos-server.odysseaqa.10.99.0.100.xip.io 10.99.0.100,10.99.0.55,10.99.0.62 80
```

### 2.2.2 Installation of the Catalogue engine

- Login to the server and go to the deployment folder

```
root@xxxxxx: cd /XXXXXXX/odyssea-deployment/k8s/cls/odyssea/
```

- Run the following command with the parameters corresponding to your case

```
root@xxxxxx:~/odyssea/odyssea-deployment/k8s/cls/odyssea# ./deploy-k8s-odyssea.sh
catalog-engine < typecluster>
```

```
Do you want to create pvc (y/n)? y
Do you want to create config map (y/n)? y
```

```
Do you want to create pods (dp files) (y/n)? y
```

```
Do you want to create services (y/n)? y
```

```
Do you want to create ingress (y/n)? y
```

```
Do you want to create cronjob (y/n)? y
```

- Below are the answers to the questions asked
- View the status of deployed pods

```
root@xxxxxx:~/odyssea/odyssea-deployment/k8s/cls/odyssea# kubectl get pods -n <typecluster>
```

- You will have to configure 52 North and GeoNetwork as well when installing from scratch

### 2.2.3 Installing the DataCollection from scratch

- Login to the server and go to the deployment folder

```
root@xxxxxx: cd /XXXXXXX/odyssea-deployment/k8s/cls/odyssea/
```

- Run the following command with the parameters corresponding to your case

Cluster	Value for the typecluster parameter
QO	odysseaqo
Production	odyssea

- Below are the answers to the questions asked

```
Do you want to create pvc (y/n)? y
```

```
Do you want to create config map (y/n)? y
```

```
Do you want to create pods (dp files) (y/n)? y
```

```
Do you want to create services (y/n)? y
```

```
Do you want to create ingress (y/n)? y
```

```
Do you want to create cronjob (y/n)? y
```

- View the status of deployed pods

```
root@xxxxxxxxxx:~/odyssea/odyssea-deployment/k8s/cls/odyssea# kubectl get pods -n <typecluster>
```

- Retrieve configurations from the following folder



```
root@xxxxxxxx:~/odyssea/odyssea-deployment/k8s/cls/odyssea#cd
/home/ubuntu/odyssea/odyssea/odyssea-deployment/k8s/cls/odyssea/data-
collection/JobConfigurations
```

- Configuring Data-collection jobs
- Check that the jobs have run

#### 2.2.4 NRPE installation from scratch

- Login to the server and go to the deployment folder

```
root@xxxxxx-dusanbe-instance1: cd /XXXXXXX/odyssea-deployment/k8s/cls/odyssea/
```

- Run the following command with the parameter corresponding to your case

```
root@xxxxxx-dusanbe-instance1:~/odyssea/odyssea-
deployment/k8s/cls/odyssea: ./deploy-nrpe-k8s-odyssea.sh < typecluster>
```

Cluster	Value for the < typecluster> parameter
QO	odysseaqo
Production	odyssea

- Below are the answers to the questions asked

```
Do you want to create pvc (y/n)? y
Do you want to create pods (dp files) (y/n)? y
Do you want to create node port access (y/n)? y
Do you want to import k8s configuration( (y/n)? n
Do you want to import nagios scripts (y/n)? n
```

- View the status of deployed pods

```
root@xxxxxx:~/odyssea/odyssea-deployment/k8s/cls/odyssea# kubectl get pods -n <
typecluster>
```

- Wait until the nrpe-XXXX pod is in 'Running' status
- Relaunch the deployment script and answer 'n' to all questions except the one asking if you want to copy the k8s configuration and scripts

```
Do you want to create pvc (y/n)? n
Do you want to create pods (dp files) (y/n)? n
Do you want to create node port access (y/n)? n
Do you want to import k8s configuration( (y/n)? y
Do you want to import nagios scripts (y/n)? y
```

- Check that the pod is running properly and that the Nagios checks are UP

### 2.2.5 Product Factory: TRIX/WavePower installation from scratch

- Connect to your deployment environment
- Go to the wps component folder

```
root@xxxxx:cd /*****/odyssea-deployment/k8s/cls/common/wps/
```

- Run the deployment script deploy-wps.sh

```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/common/wps# ./deploy-wps.sh
<prod/qo/qt>
```

- Check that the pods are running via the rancher interface or with kubectl [ **POD redis / registry / scheduler / nginxfrontal / trix /wave**]

```
root@xxxxxxx:~/odyssea/odyssea-deployment/k8s/cls/common/wps# kubectl get pods -n
odyssea

NAME READY STATUS RESTARTS
frontend-back-84d9bc5c4b-ls8c5 2/2 Running 0
frontend-odyssea-769d6674f4-57x6g 1/1 Running 0
nginxfrontal-6f7cd6c8f8-mq69f 1/1 Running 0
redis-6c65b4dcb9-sqkqq 1/1 Running 0
registry-7db66bb54f-ml82h 1/1 Running 0
scheduler-6d9bcf7d94-jqbgf 1/1 Running 0
static-data-b56889c64-r2rpj 1/1 Running 0
trix-6567d7f846-rk7ms 1/1 Running 0
wave-6c767d6c6d-7mxvm 1/1 Running 0
```

- Check that the requests are working, and the product is generated (see WPS - Script Curl page)

## 2.3 Update

### 2.3.1 Front-End Update

When a new version of the FrontEnd is received, it must be deployed on the QO (be very careful if there is an update of the **Mongo** docker). It is better to make sure that everything is working properly by letting the new version run to check that there is no impact.

- Connect to ODYSSEA's Gitlab directory
- Go to the FrontEnd registry ([https://gitshare.cls.fr/odyssea/frontend/container\\_registry](https://gitshare.cls.fr/odyssea/frontend/container_registry))
- Retrieve the latest tag version of the following images:
  - /odyssea/frontend/web
  - /odyssea/frontend/mongo
- Connect to your deployment environment
- Go to the frontend component folder

```
root@xxxxx:cd /*****/odyssea-deployment/k8s/cls/odyssea/frontend/
```

- **If there is a new version of the mongo and web images, start by updating mongoDB**
- Modify the **frontend-back-dp.yaml** deployment file by updating the version of the image retrieved from GitLab

Deployment file	Line to be updated/ checked
<b>frontend-back-dp.yaml</b>	image: <a href="#">registry-ext.cls.fr</a> :443/odyssea/frontend/mongo:vXXXX

- Deploy the frontend-back-dp.yaml file

```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/odyssea/frontend# kubectl apply -f frontend-back-dp.yaml -n <typecluster>
```

Cluster	• Value for the < typecluster> parameter
<b>QO</b>	• odysseaqo
<b>Production</b>	• odyssea

- It is possible that the base needs to be reset. A Mongo update has not yet taken place since the migration to K8S so this part of the procedure may require an update

- To update the FE you will need to modify the **frontend-dp.yaml** deployment file by updating the version of the image retrieved from GitLab

Deployment file	Line to be updated/ checked
<b>frontend-dp.yaml</b>	image: registry-ext.cls.fr:443/odyssea/frontend/web:XXXX

- Deploy the **frontend-dp.yaml** file

```
root@xxxxxx:~ /*****/odyssea-deployment/k8s/cls/odyssea/frontend# kubectl apply -f
frontend-dp.yaml -n <typecluster>
```

Cluster	Value for the < typecluster> parameter
QO	odysseaqo
Production	odyssea

- Check that the Odyssea frontend is working and that you can log in via the FE odyssea admin account

### 2.3.2 Catalogue engine update

When receiving a new version of the SOS Server, you must deploy it on the QO (be very careful if there is an update of the **postgres** docker). It is better to make sure that all are working properly by letting the new version run to check that ingestion in the DB is still possible.

- Login to ODYSSEA's Gitlab directory (Accounts and passwords )
- Go to the Catalog-engine registry ([https://gitshare.cls.fr/odyssea/sos-server/container\\_registry](https://gitshare.cls.fr/odyssea/sos-server/container_registry))
- Retrieve the latest tag version of the following images:
  - /odyssea/sos-server
  - /odyssea/sos-server/postgres
- Connect to your deployment environment
- Go to the catalog-engine component folder

```
root@xxxx:cd /*****/odyssea-deployment/k8s/cls/odyssea/catalog-engine/
```

- Modify the catalog-engine-dp.yaml deployment files by updating the image versions retrieved from GitLab

Line to be updated/ checked	Comment
image: <a href="https://registry-ext.cls.fr">registry-ext.cls.fr</a> :443/odyssea/sos-server/postgres:XXXX	Appears 2 times
image: <a href="https://registry-ext.cls.fr">registry-ext.cls.fr</a> :443/odyssea/sos-server:XXXX	

- Deploy the **catalog-engine-dp.yaml** file

```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/odyssea/catalog-engine# kubectl apply -f catalog-engine-dp.yaml -n < typecluster>
```

Cluster	Value for the typecluster parameter
<b>QO</b>	odysseaqo
<b>Production</b>	odyssea

- Check that the pods are turning well

```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/odyssea/catalog-engine# kubectl get pods -n < typecluster>
```

- Check that the interface of SOS Server works well and that you can log in with the admin account  
Check by connecting to the NRPE pod that the Export folder contains files (see procedure in the Visualization of the log files)
- If ok, run an ingestion with the cron job

```
• root@xxxxxxxxxxxxxxxxx/*****/odyssea-deployment/k8s/cls/odyssea/catalog-engine: kubectl create job --from=cronjob/catalog-engine-cronjob < name_in_choice> -n < typecluster>
```

- Check from the NRPE pod that the **/opt/Processed** folder is updated with new files

### 2.3.3 Updating the DataCollection

When receiving a new version of the DataCollection, you must first deploy it to the QO. Make sure by letting the new version run that it does not impact the data recovery or the ingestion in the DB

- Connect to ODYSSEA's Gitlab directory
- Go to the dataCollection registry ([https://gitshare.cls.fr/odyssea/data-collection/container\\_registry](https://gitshare.cls.fr/odyssea/data-collection/container_registry))
- Retrieve the latest tag version of the following images:
  - /odyssea/data-collection/c7.db
  - /odyssea/data-collection/c7.dwd

- odyssea/data-collection/c7.api.ui
- Connect to your deployment environment
- Go to the data-collection component folder

```
• root@xxxxxxx:cd /*****/odyssea-deployment/k8s/cls/odyssea/data-collection/
```

- Modify the deployment files **data-collection-hangfire-dp.yaml/data-collection-dp.yaml** by updating the image versions retrieved from GitLab

Deployment file	Line to be updated/ checked	Comment
<b>data-collection-dp.yaml</b>	image: <a href="#">registry-ext.cls.fr</a> :443/odyssea/data-collection/c7.api.ui:XXX	
<b>data-collection-hangfire-dp.yaml</b>	image: <a href="#">registry-ext.cls.fr</a> :443/odyssea/data-collection/c7.db:XXX	Appears 2 times
	image: <a href="#">registry-ext.cls.fr</a> :443/odyssea/data-collection/c7.dwd:XXX	

- Deploy the **data-collection-hangfire-dp.yaml** file

```
root@xxxxxxx:~ /*****/odyssea-deployment/k8s/cls/odyssea/data-collection# kubectl
apply -f data-collection-hangfire-dp.yaml -n < typecluster>
```

Cluster	Value for the < typecluster> parameter
<b>QO</b>	odysseaqo
<b>Production</b>	odyssea

- Deploy the **data-collection-dp.yaml** file

```
root@xxxxxxx:~ /*****/odyssea-deployment/k8s/cls/odyssea/data-collection# kubectl
apply -f data-collection-dp.yaml -n < typecluster>
```

Cluster	Value for the typecluster parameter
<b>QO</b>	odysseaqo
<b>Production</b>	odyssea

- Check that the pods are turning well

```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/odyssea/data-collection# kubectl
get pods -n <typecluster>
```

- Check that the DataCollection ingress is working (hangfire/swagger/job management interface)
- Verify that the job configuration interface is working by running a file recovery job
- Check by connecting to the NRPE pod that the Export folder has been populated (see procedure in [Viewing log files](#))
- If ok, run an ingestion with the cron job

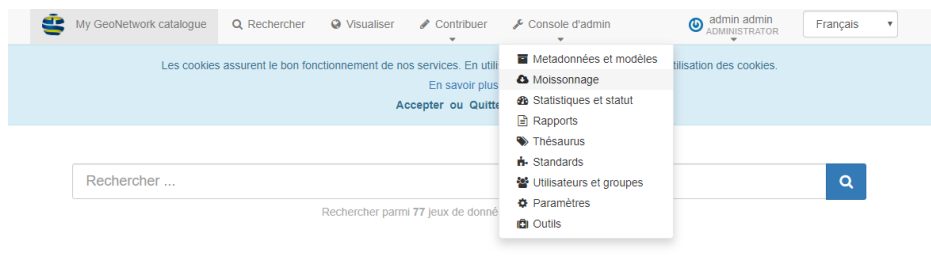
```
root@xxxxxxx:~/*****/odyssea-deployment/k8s/cls/odyssea/data-collection# kubectl
create job --from=cronjob/catalog-engine-cronjob <name_at_choice> -n <typecluster>
```

- Check from the NRPE pod that the /opt/Processed folder is updated with new files

## 2.4 Configuration

### 2.4.1 Configuration of GeoNetwork

- Connect to GeoNetwork
- Go to Admin Console → Harvesting and fill in the items below to retrieve data from CMEMS and the Datastore



- Create a CSW catalog for the CLS DATASTORE, with the following parameters:
  - Node name and logo: CLS (datastore)
  - Frequency: 0 0 12 \* \* ?
  - Service URL: <https://datastore.cls.fr/csw/>
  - Apiso:Any Text→ projects:ODYSSEA or (Apiso:Any Text → free what we put in prod to get HYCOM data too)
  - Privileges: All
- Create a GeoNetwork catalog for CMEMS
  - Node name; CMEMS
  - Frequency: 0 0 12 \* \* ?
  - Catalogue URL: <https://cmems-catalog-ro.cls.fr/geonetwork/>
  - geonetwork-nodeHelp: srv

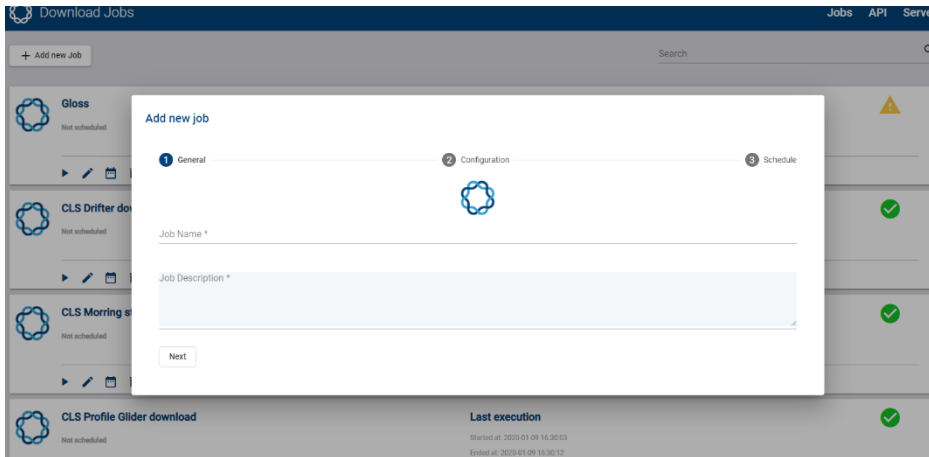
- Search filter (Title): \*MED\*
- Privileges: All

## 2.4.2 Datacollection Job Configuration

- Connect to your deployment server
- Get the JSONs of the jobs you want to add from the configuration folder available on the Odyssea repo

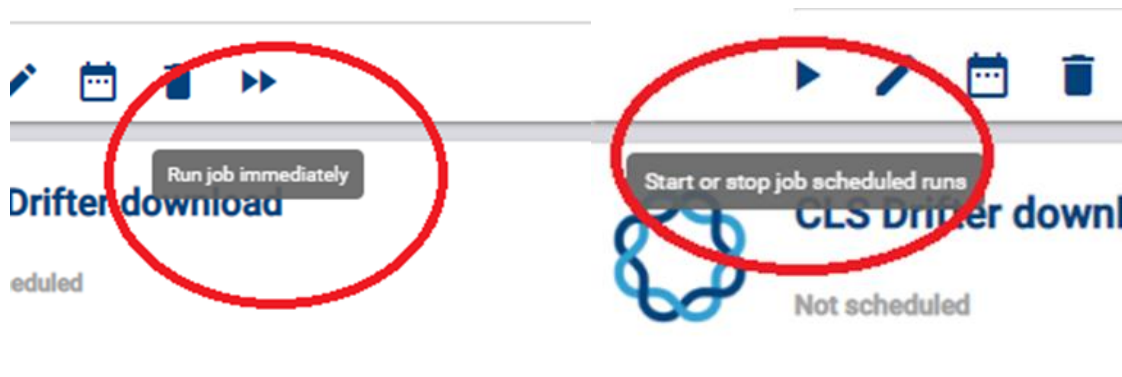
```
root@xxxxxxxx: cd /home/ubuntu/odyssea/odyssea-deployment/k8s/cls/common/data-collection/JobConfigurations
```

- Then go to the DataCollection jobs interface ([Accounts and passwords](#))
- Click on the **Add new job** tab which will open the window below



- In the **General** tab add the name and a description
- Click on **Next** and copy/paste the content of the JSON file for the job in question
- Click on **Next** to define the job scheduling ( \*/30 \* \* \* \* preferably)
- Press **Create Job** and the job should appear in the list of configured jobs
- Run the job immediately to see if it works





- If ok, don't forget to program the scheduling by pressing Play

### 2.4.3 MongoDB configuration

#### 2.4.3.1 Managing user rights with MongoDB

- Connect to the mongoDB pod

```
ubuntu@xxxxxxx:~$ kubectl exec -it <frontend-back-XXXX> -c mongo bash
```

- Connecting to mongo client

```
root@frontend-back-65fdd4bd9d-429nd:/db# mongo -u admin -p UDW6cDyS7hm2xHrF --authenticationDatabase odysseaplatform
```

- Connection to the DB

```
> use odysseaplatform
```

- Update a user

```
> db.users.find({"local.email": "XXXXXX"}).pretty()
> db.users.updateOne({"_id": ObjectId("5cc6fbafe9cc85001465318e")},{$set:{"is_admin":true,"is_superuser":true}});
```

- Delete a user

```
> db.users.remove({"_id": ObjectId("5cc6fbafe9cc85001465318e")});
```

- Other useful commands, remember to connect to the mongo DB before doing any of the commands below

```
> db.users.find({"local.email": "XXXX"}).pretty()
> db.users.find({}).pretty()
> db.users.find({"is_admin":true}).pretty()
```

```
> db.users.find({"is_superadmin":true}).pretty()

>      db.users.updateOne({"_id"      :      ObjectId("xxxxx")},{ $set:{"is_admin":true,
"is_superadmin":true}});

> db.users.insert({"_id" : ObjectId("xxxxxx"), "is_admin":true, "is_superadmin":true}) --> new
line (1 new document in the "user" collection, in
assuming the ID does not exist. Otherwise error)
```

#### 2.4.3.2 Create a dump of the Mongo DB

- Connect to the mongoDB pod

```
ubuntu@sel-mokhtari-ramus-instance1:~$ kubectl exec -it <frontend-back-XXXX> -n
<namespace> -c mongo bash
```

- Dump creation

```
root@frontend-back-d957f79b4-6xpl8:/db# mongodump --username admin --password
UDW6cDyS7hm2xHrF -d odysseaplatform --out <dump name of choice>
```

#### 2.4.3.3 Create a dump of the Mongo DB

- Connecting to the mongoDB pod

```
ubuntu@sel-mokhtari-ramus-instance1:~$ kubectl exec -it <frontend-back-XXXX> -n
<namespace> -c mongo bash
```

- Connecting to mongo client

```
root@frontend-back-65fdd4bd9d-429nd:/db# mongo -u admin -p UDW6cDyS7hm2xHrF --
authenticationDatabase odysseaplatform
```

- Connection to BDD

```
> use odysseaplatform
```

- Deleting the DB

```
> db.dropDatabase()

> exit
```

- Catering from the dump of your choice

```
root@xxxxxxx:~/odyssea/odyssea-deployment# kubectl exec -it frontend-back-XXXXXXX -c
mongo bash -n <typecluster>
```

```
root@frontend-back-5f6c7cbcd4-mx6g7:/db: mongo admin --username admin --password
UDW6cDyS7hm2xHrF < script.js

root@frontend-back-5f6c7cbcd4-mx6g7:/db: mongorestore --username admin --password
"UDW6cDyS7hm2xHrF" -d odysseaplatform "mongodump-2020-MM-DD/odysseaplatform/"
```

#### 2.4.4 Product Factory Configuration

This section will detail the procedure to follow in order to launch scripts that manage the jobs that generate the netCDF files from the TPIX or WavePower algorithm.

The current configuration allows the generation of 4 files per day (D to D+3).

There are 4 scripts that will allow you to restart the generation of netCDF files. They are available in the GIT Repository of curl scripts

##### 2.4.4.1 Script execution of daily jobs

The `execute_job_trix.sh` or `execute_job_wavepower.sh` script (depending on the product you want to replay) will allow you to set up a job that will run once a day with no arguments to provide.

```
root@xxxxx:~/odyssea/odyssea-deployment/k8s/cls/common/deploy/wpsjobs# ./execute_job_trix.sh

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<
      wps:StatusInfo
      xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"      xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">

  < wps:JobID>ALGO_SCHEDULER_TRIX_TRIX_2020-04-27_13_22_46_773</wps:JobID>

  < wps:Status>Accepted</wps:Status>

I'm not sure if I'm going to be able to do that.
```

If you need to replay the trix / wavepower algo for a previous day you need to modify the `execute_job_<trix or wavepower>.sh` script by replacing the `${date}` tag with YYYY-MM-DD which is the date of the run you want to replay

```
< wps:Input id="startDate">
  < wps:Data> ${date}</wps:Data>
```

I'm not sure how to do that.

```
< wps:Input id="startDate">
  < wps:Data> YYYY-MM-DD</wps:Data>
```

I'm not sure how to do that.

#### 2.4.4.2 Script tracking logs

The log.sh script allows to follow the execution of a job and to check if the generation of the files is done without errors.

You must provide this script with the id of the job that will be returned in response to the request (in the example above the job id is ALGO\_SCHEDULER\_TRIX\_20-04-27\_13\_22\_46\_773).

```
root@sel-mokhtari-ramus-instance1:~/odyssea/odyssea-deployment/k8s/cls/common/deploy/wpsjobs#./job_log.sh
ALGO_SCHEDULER_TRIX_2020-04-27_13_22_46_773

< wps:Result      xmlns:wps="http://www.opengis.net/wps/2.0"      xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows/2.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">

  < wps:Output id="Log lines">

    < wps:Data mimeType="text/plain">2020-04-27T13:22:46.793 INFO Launching execution Thread for algo=trix@trix
with 1 run(s) by execution.

2020-04-27T13:22:47.091 INFO Received StatusInfo ALGO_TRIX_2020-04-27_13_22_46_982 for JobId 'Accepted'

2020-04-27T13:22:47.092INFO Sleeping 30000 ms until next polling status for JobId ALGO_TRIX_2020-04-27_13_22_46_982

2020-04-27T13:23:17.093      INFO      Sending      command      getStatus      on      http://service-
nginxfrontal.odysseaprod.svc.cluster.local:9090/wps/

2020-04-27T13:23:17.199      INFO      StatusInfo      response      received      for      getStatus      on      http://service-
nginxfrontal.odysseaprod.svc.cluster.local:9090/wps/

2020-04-27T13:23:17.200 INFO Job Status is Running

.....

.....
```

#### 2.4.4.3 Script listing the jobs

The job\_list.sh script is used to retrieve the list of jobs that are executed. The launch of this script does not require to provide any parameters.

```

root@sel-mokhtari-ramus-instance1:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs# ./job_list.sh

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<      wps:Result      xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"      xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">

    < wps:Output id="Number of scheduled job(s)">

        < wps:Data mimeType="text/plain">1</wps:Data>

        I'm not sure how to do that.

    < wps:Output id="Scheduled job identifier">

        <      wps:Data      mimeType="text/plain">ALGO_SCHEDULER_TRIX_TRIX_2020-04-
27_14_52_38_474</wps:Data>

        I'm not sure how to do that.

</wps:Result>

```

#### 2.4.4.4 Script listing the jobs

The `dismiss_job.sh` script allows you to delete a job. To do so, you just must provide the job ID as an argument (you can find it thanks to the `job_list.sh` script for example).

```

root@sel-mokhtari-ramus-instance1:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs# ./dismiss_job.sh ALGO_SCHEDULER_TRIX_TRIX_2020-04-
27_14_52_38_474

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<      wps:StatusInfo      xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"      xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd">

    < wps:JobID>ALGO_SCHEDULER_TRIX_TRIX_2020-04-27_14_52_38_474</wps:JobID>

    < wps:Status>Dismissed</wps:Status>

    I'm not sure if I'm going to be able to do that.

```

#### 2.4.4.5 host, hardware settings

OS target: Linux 64bits (Tested on centos-7.2.1511)

Minimal configuration for an operational usage:

- CPU: 4 CPU, 2,4GHz
- RAM: 32 Gb RAM
- Storage:
  - Motu installation folder 15Gb: can be installed on the OS partition (default folder /opt/cmems-cis)
  - Motu download folder 200Gb: by default, /opt/cmems-cis/motu/data/public/download
  - Has to be installed in a dedicated partition to avoid freezing Motu if disk is full. Note that the available space of the download folder has to be tuned, depending on:
    - The number of users which run requests at the same time on the server
    - The size of the data distributed.

Once started, you can check its performance.

For test usage we recommend:

- CPU: 2 CPU, 2,4GHz
- RAM: 10 Gb RAM
- Storage:
  - Motu installation folder 15Gb
  - Motu download folder 50Gb: by default, motu/data/public/download

#### 2.4.4.6 Motu host, software settings

Motu embeds all its dependencies (Java , Tomcat, CDO). All versions of these dependencies will be visible in the folder name once the Motu product archive is extracted.

For example:

```
ls -l /opt/cmems-cis/motu/products
apache-tomcat-7.0.69
cdo-group
jdk1.7.0_79
README
version-products.txt
```

So, bash shell is only required on the Linux host machine.

#### 2.4.4.7 External interfaces

Motu is able to communicate with different external servers:

- Unidata | THREDDS Data Server (TDS): Motu has been only tested with TDS v4.6.14 2019-07-29. The links to this server are set in the Business settings and are used to run OpenDap or subsetter interfaces. If Motu runs only with DGF, this server is not required.

Note that some specific characters have to be relaxed, e.g. when TDS is installed on Apache Tomcat, add attribute `relaxedQueryChars="<>[]{}"` in the connector node by editing `conf/server.xml` from your TDS tomcat installation folder:

```
<Connector relaxedQueryChars="&lt;&gt;[]{}" port="8080" ...
```

as reported in this forum topic.

Without this configuration Motu server can raised exceptions visible in the Motu "errors.log", e.g.:

```
ERROR fr.cls.atoll.motu.web.bll.catalog.product.cache.CacheUpdateService.updateConfigService Error
during refresh of the describe product cache, config service=..., productId=...

fr.cls.atoll.motu.web.bll.exception.MotuException: Error in NetCdfReader open - Unable to aquire
dataset - location data:

Caused by: java.io.IOException: http://.../thredds/dodsC/$dataset is not a valid URL, return status=400
```

Single Sign-On - CAS: The link to this server is set in the System settings. If Motu does not use SSO, this server is not required.

The installation of these two servers is not detailed in this document. Refer to their official web site to know how to install them.

#### 2.4.4.8 Several Motu instances on a same host

If you need to instance several instances of Motu server on a same host, you have to:

- RAM: set 32Go of RAM for each instance. For example, two Motu instances on a same host requires 64Go
- Storage: allocate disk space, in relationship with the Motu usage. Download dedicated partition can be shared or dedicated.
- Folders: Install each Motu instance in a dedicated folder:
  - /opt/motu1/motu,
  - /opt/motu2/motu,
  - ...,
  - /opt/motuN/motu

#### 2.4.5 Install Motu from scratch

Motu installation needs two main steps: the software installation and optionally the theme installation.

The software installation brings by default the CLS theme. The theme installation is there to customize or change this default theme.

#### 2.4.5.1 Install Motu software, for example on a Dissemination Unit

Copy the installation archives and extract them.

```
cd /opt/cmems-cis
cp motu-products-A.B.tar.gz .
cp motu-distribution-X.Y.Z.tar.gz .
cp motu-config-X.Y.Z-$BUILDID-$TARGET-$PLATFORM.tar.gz .
tar xzf motu-products-A.B.tar.gz
tar xzf motu-distribution-X.Y.Z.tar.gz
tar xzf motu-config-X.Y.Z-$BUILDID-$TARGET-$PLATFORM.tar.gz
cd motu
```

At this step, Motu is able to start. But static files used for customizing the web theme can be installed.

In the CMEMS context, the installation on a dissemination unit is ended, static files are installed on a central server.

Now you can configure the server:

- Set the system properties: http port, ...
- Configure dataset
- Configure the logs

Refer to configuration in order to check your configuration settings.

Motu is installed and configured. You can start Motu server.

Then you can check installation.

#### 2.4.5.2 Install Motu theme (public static files)

As a dissemination unit administrator, in CMEMS context, this section is not applicable.

Public static files are used to customized Motu theme. When several Motu are installed, a central server eases the installation and the update by referencing static files only once on a unique machine. This is the case in the CMEMS context, where each dissemination unit host a Motu server, and a central server hosts static files.



If you run only one install of Motu, you can install static files directly on Motu Apache tomcat server.

### 2.4.5.3 On a central server

Extract this archive on a server.

```
tar xvzf motu-web-static-files-X.Y.Z-classifier-$timestamp-$target.tar.gz
```

Then use a server to make these extracted folders and files accessible from an HTTP address.

Example: The archive contains a motu folder at its root. Then a particular file is "motu/css/motu/motu.css" and this file is served by the URL <http://resources.myocean.eu/motu/css/motu/motu.css> in the CMEMS CIS context.

#### 2.4.5.3.1 DIRECTLY ON MOTU APACHE TOMCAT SERVER

If you do not use a central entity to serve public static files, you can optionally extract the archive and serve files directly by configuring Motu.

First extract the archive:

```
tar xzf motu-web-static-files-X.Y.Z-classifier-$timestamp-$target.tar.gz -C /opt/cmems-cis/motu/data/public/static-files
```

Then edit "motu/tomcat-motu/conf/server.xml" in order to serve files from Motu.

Add then "Context" node as shown below. Note that several "Context" nodes can be declared under the Host node.

```
[...]  
<Host appBase="webapps" [...]  
    <!-- Used to serve public static files -->  
    <Context docBase="/opt/cmems-cis/motu/data/public/static-files/motu" path="/motu"/>  
[...]
```

Finally, in motuConfiguration.xml, remove all occurrences of the attribute named: httpBaseRef in motuConfig and configService nodes (Do not set it empty, remove it).

If you want to set another path instead of "/motu", you have to set also the business configuration parameter named httpBaseRef.

#### 2.4.5.4 Check installation

##### 2.4.5.4.1 START MOTU

```
./motu start
```

##### 2.4.5.4.2 CHECK MESSAGES ON THE SERVER CONSOLE

When you start Motu, the only message shall be:

```
tomcat-motu - start
```

Optionally, when this is your first installation or when a software update is done, an INFO message is displayed:

```
INFO: War updated: tomcat-motu/webapps/motu-web.war [$version]
```

If any other messages appear, you have to treat them.

As Motu relies on binary software like CDO, error could be raised meaning that CDO does not runs well.

```
ERROR: cdo tool does not run well: $cdo --version  
[...]
```

In this case, you have to install CDO manually.

##### 2.4.5.4.3 CHECK MOTU WEB SITE AVAILABLE

Open a Web browser, and enter: `http://$motuUrl/motu-web/Motu?action=ping`

Where \$motuUrl is: ip adress of the server:tomcat port Refer to configuration regarding the tomcat port.

Response has to be:

```
OK - response action=ping
```

Open a Web browser, and enter: `http://$motuUrl/motu-web/Motu`.

If nothing appears, it is because you have to add a dataset.

##### 2.4.5.4.4 CHECK MOTU LOGS

Check that no error appears in Motu errors log files.

#### 2.4.6 Motu Configuration

This chapter describes the Motu configuration settings.

All the configuration files are set in the \$installDir/motu/config folder.

- Configuration directory structure

- Business settings
- System settings
- Log settings

#### 2.4.6.1 Configuration directory structure

cd \$installDir/motu/config

- config: Folder which contains the motu configuration files.
  - motu.properties: JVM memory, network ports of JVM (JMX, Debug) and Tomcat (HTTP, HTTPS, AJP, SHUTDOWN). CAS SSO server settings.
  - motuConfiguration.xml: Motu settings (Service, Catalog via Thredds, Proxy, Queues, ....)
  - log4j.xml: Log4j v2 configuration file
  - standardNames.xml: Contains the standard names
  - version-configuration.txt: Contains the version of the current Motu configuration.

#### 2.4.6.2 Business settings

##### 2.4.6.2.1 MOTUCONFIGURATION.XML: MOTU BUSINESS SETTINGS

This file is watched and updated automatically. This means that when Motu is running, this file has to be written in an atomic way.

You can configure 3 main categories:

- MotuConfig node : general settings
- ConfigService node : catalog settings
- QueueServerConfig node : request queue settings
- RedisConfig node : Redis server config

If you have not this file, you can extract it (set the good version motu-web-X.Y.Z.jar):

```
/opt/cmems-cis/motu/products/jdk1.7.0_79/bin/jar      xf      /opt/cmems-cis/motu/tomcat-  
motu/webapps/motu-web/WEB-INF/lib/motu-web-X.Y.Z.jar motuConfiguration.xml
```

If you have this file from a version anterior to Motu v3.x, you can reuse it. In order to improve global performance, you have to upgrade some fields:

- ncscs Set it to "enabled" to use a faster protocol named sub-setter rather than OpenDap to communicate with TDS server. ncscs must be enabled only with regular grid. The datasets using curvilinear coordinates (like ORCA grid) cannot be published with ncscs. Thus, ncscs option must be set to disable or empty.
- httpBaseRef shall be set to the ULR of the central repository to display the new theme
- ExtractionFilePatterns to give a custom name to the downloaded dataset file

#### 2.4.6.2.2 ATTRIBUTES DEFINED IN MOTUCONFIG NODE

- **defaultService**

A string representing the default action in the URL /Motu?action=\$defaultService.

The default one is "listservices".

All values can be found in the method `USLRequestManager#onNewRequest` with the different ACTION\_NAME.

- **dataBlockSize**

Amount of data in Ko that can be requested in a single query from Motu to TDS. Default is 2048Kb.

If this amount is lower than the `maxSizePerFile` (in MegaBytes), Motu will launch several sub-requests to TDS to gather all the data.

Higher value (up to the `maxSizePerFile`) leads to less requests, but with higher data volume to transfer by request from TDS to Motu. And the `tds.http.sotimeout` has to be long enough for letting TDS the time to read and transfer the whole data to Motu.

Lower values will imply more requests, but shorter. It consumes more CPU on both Motu and TDS with the advantages to allow TDS to answer to other parallel request it would receive, and to reduce communication times for each request.

Concerning performance, we tried 200Mb and 1024Mb for a 1024Mb request, and durations were similar, but note that the shapes of the sub-requests may not match the shapes of the netcdf files, and that could imply a supplementary delay for hard drive data storage.

*If the `tds.http.sotimeout` can be set to a high value (such as 900s for a 1Gb max size request), the safest is to use the `maxSizePerFile` value for the `dataBlockSize` parameter (mind the units Kb/Mb). Else from the max timeout the environment can support (external constraints or "004-27" error, see `tds.http.sotimeout`), extrapolate a volume of data that can be transferred during this delay, lower it to keep a margin, and use it as the `dataBlockSize`.*

- **maxSizePerFile**

This parameter is only used with a catalog type set to "FILE" meaning a DGF access.

It allows download requests to be executed only if data extraction is lower than this parameter value.

Unit of this value is MegaBytes.

Default is 1024 MegaBytes.

Example: `maxSizePerFile="2048"` to limit a request result file size to 2GB.

- **maxSizePerFileSub**

This parameter is only used with a catalog type used with Opendap or Ncss.

It allows download requests to be executed only if data extraction is lower than this parameter value.

Unit of this value is MegaBytes.

Default is 1024 MegaBytes.

Example: `maxSizePerFileSub="2048"` to limit request result file size to 2GB.

- **maxSizePerFileTDS**

@Deprecated from v3 This parameter is not used and has been replaced by `maxSizePerFile` and `maxSizePerFileSub`.

Number of data in Megabytes that can be written and download for a Netcdf file. Default is 1024Mb.

- **extractionPath**

The absolute path where files downloaded from TDS are stored.

For example: `/opt/cmems-cis/motu/data/public/download` It is recommended to set this folder on a hard drive with very good performances in write mode. It is recommended to have a dedicated partition disk to avoid freezing Motu if the hard drive is full. By default, value is `$MOTU_HOME/data/public/download`, this folder can be a symbolic link to another folder.

String with format `${var}` will be substituted with Java property variables. @See `System.getProperty(var)`.

- **downloadHttpUrl**

Http URL used to download files stored in the "extractionPath" described above. It is used to allow users to download the result data files.

This URL is concatenated to the result data file name found in the folder "extractionPath".

When a frontal HTTPd server is used, it is this URL that shall be configured to access to the folder "extractionPath".

String with format `${var}` will be substituted with Java property variables. @See `System.getProperty(var)`.

- **httpBaseRef**

Http URL used to serve files from to the path where archive `motu-web-static-files-X.Y.Z-classifier-buildId.tar.gz` has been extracted.

For example:

- When `httpBaseRef` is set to an URL, for example `"http://resources.myocean.eu/motu"`, this URL serves a folder which contains `./css/motu/motu.css`.
- For example, it enables to serve the file `http://resources.myocean.eu/motu/css/motu/motu.css`.
  - When `httpBaseRef` is set to `"."`, it serves static files which are included by default in Motu application
  - When `httpBaseRef` is removed (not just empty but attribute is removed), it serves a path accessible from URL `$motuIP/${motuContext}/motu`

**IMPORTANT:** When Motu URL starts with "HTTPS", if you set an URL in httpBaseRef, this URL has also to start with "HTTPS". On the contrary, when Motu URL starts with "HTTP", if you set an URL in httpBaseRef, this URL can start with "HTTP" or "HTTPS".

- **cleanExtractionFileInterval**

In minutes, oldest result files from extraction request which are stored in the folder set by extractionpath are deleted. This check is done each "runCleanInterval" minutes.

Default = 60min

- **cleanRequestInterval**

In minutes, oldest status (visible in debug view) than this time are removed from Motu. This check is done each "runCleanInterval" minutes.

Default = 60min

- **runCleanInterval**

In minutes, the waiting time between each clean process. The first clean work is triggered when Motu starts.

A clean process does:

- delete files inside java.io.tmpdir
- delete all files found in extractionFolder bigger than extractionFileCacheSize is Mb
- delete all files found in extractionFolder oldest than cleanExtractionFileInterval minutes
- remove all status oldest than cleanRequestInterval minutes

Default = 1min

- **extractionFilePatterns**

Patterns (as regular expression) that match extraction file name to delete in folders:

- java.io.tmpdir
- extractionPath

Default is "..nc\$|..zip\$|..tar\$|..gz\$|.\*.extract\$"

- **extractionFileCacheSize**

Size in Mbytes.

A clean job runs each "runCleanInterval". All files with a size higher than this value are deleted by this job. If value is zero, files are not deleted.

Default value = 0.

- **describeProductCacheRefreshInMilliSec**

Provide the delay to wait to refresh the meta-data of products cache after the last refresh.

Motu has a cache which is refreshed asynchronously. Cache is first refreshed as soon as Motu starts.

Then Motu waits for this delay before refreshing again the cache.

This delay is provided in millisecond.

The default value is 60000 meaning 1 minute.

Logbook file (motu/log/logbook.log) gives details about time taken to refresh cache, for example:

```
INFO  CatalogAndProductCacheRefreshThread.runProcess Product and catalog caches refreshed in
2min 19sec 75msec
```

Logbook file gives details per config service (\$configServiceId) about dedicated time taken to refresh cache, for example:

```
INFO          CatalogAndProductCacheRefreshThread.runProcess      Refreshed      statistics:
$configServiceId@Index=0min 34sec 180msec, $configServiceId@Index=0min 31sec 46msec, ...
```

They are sorted by config service which has taken the most time first.

@Index All config services are refreshed sequentially. This index is the sequence number for which this cached has been refreshed.

Example of archived data with several TB of data. Cache is refreshed daily:  
describeProductCacheRefreshInMilliSec=86400000

Example of real time data with several GB of data. Cache is refreshed each minute:  
describeProductCacheRefreshInMilliSec=60000

- **runGCInterval**

@Deprecated from v3 This parameter is not used.

- **httpDocumentRoot**

@Deprecated from v3 This parameter is not used. Document root of the servlet server.

- **wcsDcpUrl**

Optional attribute. Used to set the tag value "DCP" in the response of the WCS GetCapabilities request with a full URL. The WCS DCP URL value is define using the following priority order: - The value of this parameter defines on the motuConfiguration.xml file. The value can be directly the URL to use or the name of a java property define between {} which contains the value of the URL. - The java property "wcs-dcp-url" value - The URL of the web server on which Motu webapps is deployed. This attribute can be set when you use a frontal web server to serve the WCS requests, e.g., http://myFrontalWebServer/motu/wcs and your frontal is an HTTP proxy to http://motuWebServer/motu-web/wcs.

- **useAuthentication**

@Deprecated from v3 This parameter is not used. It is redundant with parameter config/motu.properties#cas-activated.

- **defaultActionIsListServices**

@Deprecated from v3 This parameter is not used.

- **Configure the Proxy settings**

@Deprecated from v3 This parameter is not used. To use a proxy in order to access to a Threads, use the JVM properties, for example:

```
tomcat-motu-jvm-javaOpts=-server -Xmx4096M ... -Dhttp.proxyHost=monProxy.host.fr -
Dhttp.proxyPort=XXXX -Dhttp.nonProxyHosts='localhost|127.0.0.1'
```

- useProxy
- proxyHost
- proxyPort
- proxyLogin
- proxyPwd

- **refreshCacheToken**

This token is a key value which is checked to authorize the execution of the cache refresh when it is request by the administrator. If the token value provided by the administrator doesn't match the configured token value, the refresh is not executed and an error is returned. A default value "a7de6d69afa2111e7fa7038a0e89f7e2" is configured but it's hardly recommended to change this value. If this token is not changed, it is a security breach and a log ERROR will be written while the configuration will be loaded. The value can contains the characters [A-Za-z] and specials listed here ( -\_@\$\*!;.,?()[] ) It's recommended to configure a token with a length of 29 characters minimum.

- **downloadFileNameFormat**

Format of the file name result of a download request.

2 dynamic parameters can be used to configure this attribute:

- @@requestId@@: this pattern will be replaced in the final file name by the id of the request.
- @@productId@@: this pattern will be replaced in the final file name by the id of the requested product.

If this attribute is not present, default value is: "@@productId@@\_@@requestId@@.nc"

- **motuConfigReload**

Configure how motu configuration is reloaded.

Arguments are only 'inotify' or an 'integer in seconds'. 'inotify' is the default value.

- 'inotify': reload as soon as the file is updated (works only on local file system, not for NFS file system).
- 'integer in seconds': reload each X second the configuration in 'polling' mode. If this integer is equals or lower than 0, it disables the refresh of the configuration.



- **Attributes defined in configService node**

**name**

String to set the config service name. If the value of this attribute contains some special characters, those characters have not to be encoded. For example, if the value is an URL, the characters ":" and "/" have not to be encoded like "%2E" or "%3A".

**group**

String which describes the group

**description**

String which describes the service

**profiles**

Optional string containing one value, several values separated by a comma or empty (meaning everybody can access).

Used to manage access right from a SSO cas server.

In the frame of CMEMS, three profiles exist:

- internal: internal users of the CMEMS project
- major: major accounts
- external: external users

Otherwise, it's possible to configure as many profiles as needed.

Profiles are configured in LDAP within the attribute "memberUid" of each user. This attribute is read by CAS and is sent to Motu once a user is logged in, in order to check if it matches profiles configured in Motu to allow a user accessing the data.

In LDAP, "memberUid" attribute can be empty, contains one value or several values separated by a comma.

**velocityTemplatePrefix**

Optional, string used to target the default velocity template. It is used to set a specific theme.

Value is the velocity template file name without the extension.

Default value is "index".

**refreshCacheAutomaticallyEnabled**

Optional, Boolean used to determine if the current config service has its cache updated automatically by Motu or not. Default value is "true". "true" means that the config service cache update is executed automatically by Motu.

**httpBaseRef**

Optional, used to override motuConfig httpBaseRef attribute for this specific service.

**defaultLanguage**

@Deprecated from v3 This parameter is not used.

- Attributes defined in catalog node

**name**

This catalog name refers a TDS catalog name available from the URL: `http://$ip:$port/thredds/m_HR_MOD.xml` Example: `m_HR_OBS.xml`

**type**

- tds: Dataset is downloaded from TDS server. In this case, you can use Opendap or NCSS protocol.
- file: Dataset is downloaded from DGF

Example: tds

**ncss**

Optional parameter used to enable or disable the use of NetCDF Subset Service (NCSS) in order to request the TDS server. ncss must be enabled only with regular grid. The datasets using curvilinear coordinates (like ORCA grid) cannot be published with ncss. Thus, ncss option must be set to disable or empty. Without this attribute or when empty, Motu connects to TDS with Opendap protocol. If this attribute is set to "enabled" Motu connects to TDS with NCSS protocol in order to improve performance.

We recommend to use "enabled" for regular grid datasets. Values are: "enabled", "disabled" or empty.

**urlSite**

- TDS URL

For example: `http://$ip:$port/thredds/`

- DGF URL

For example: `file:///opt/publication/inventories`

- **Attributes defined in queueServerConfig node**

**maxPoolAnonymous**

Maximum number of request that an anonymous user can send to Motu before throwing an error message.

Value of -1 means that no check is done, so an unlimited number of users can request the server.

Default value is 10.

In case where an SSO server is used for authentication, this parameter is not used. In this case you will be able to fix a limit by setting "maxPoolAuth" parameter value.

**maxPoolAuth**

Maximum number of request that an authenticated user can send to Motu before throwing an error message.

Value of -1 means that no check is done, so an unlimited number of users can request the server.

Default value is 1.

In case where no SSO server is used for authentication, this parameter is not used. In this case you will be able to fix a limit by setting "maxPoolAnonymous" parameter value.

### **defaultPriority**

@Deprecated from v3. This parameter is not used.

- Attributes defined in queues

### **id**

An id to identify the queue.

### **description**

Description of the queue.

### **batch**

@Deprecated from v3 This parameter is not used.

### **Child node: maxThreads**

Use to build a `java.util.concurrent.ThreadPoolExecutor` to set "corePoolSize" and "maximumPoolSize" values.

Default value is 1.

The total number of threads should not be up to the total number of cores of the processor on which the Motu is running.

### **Child node: maxPoolSize**

Request are put in a queue before being executed by the `ThreadPoolExecutor`. Before being put in the queue, the queue size is checked. If it is upper than this value `maxPoolSize`, an error message is returned. Value of -1 means no check is done.

Default value is -1

### **Child node: dataThreshold**

Size in Megabyte. A request has a size. The queue in which this request will be processed is defined by the request size. All queues are sorted by size ascending. A request is put in the last queue which has a size lower than the requested size. If the requested size is higher than the bigger queue `dataThreshold`, request is not treated and an error message is returned.

This parameter is really useful when a Motu is used to server several kinds of file size and you want to be sure that file with a specific size does no slow down the request for small data size.

In this case, you can configure two queues and set up a number of threads for each in order to match the number of processors. The JVM, even if requests for high volume are running, will be able to process smallest requests by running the thread on the other processor core. Sp processing high volume requests will not block the smallest requests.

#### **Child node: lowPriorityWaiting**

@Deprecated from v3 This parameter is not used.

- Attributes defined in redisConfig node

This optional node is used to run Motu in a scalable architecture. Do not add this node when you just run one single Motu instance.

Once this node is added, Motu stores all its request ids and status in Redis.

#### **host**

Define the host (ip or server name) where is deployed the Redis server od Redis cluster used by Motu to share the RequestId and RequestStatus data. Default value is localhost.

#### **port**

Define the port used by the Redis server or Redis cluster used by Motu to share the requestId and RequestStatus data. Default value is 6379.

#### **prefix**

Define the prefix used to build the RequestId value of the shared RequestStatus data. Default value is requestStatus.

#### **isRedisCluster**

Define if the redis server in cluster mode. This is a boolean value. By default, is set to FALSE and the cluster mode is not activated. To activate the cluster, the value has to be set on TRUE.

### **2.4.6.3 System settings**

#### **2.4.6.3.1 MOTU.PROPERTIES: MOTU SYSTEM SETTINGS**

System settings are configured in file config/motu.properties

All parameters can be updated in the file.

- Java options
- Tomcat network ports
- CAS SSO server

#### 2.4.6.3.2 JAVA OPTIONS

The three parameters below are used to tune the Java Virtual Machine, and the tomcat-motu-jvm-javaOpts parameter can include any Java property in the form "-D<java property name>=<value>":

# -server: tells the Hostspot compiler to run the JVM in "server" mode (for performance)

```
tomcat-motu-jvm-javaOpts=-server -Xmx4096M -Xms512M -XX:MetaspaceSize=128M -  
XX:MaxMetaspaceSize=512M
```

```
tomcat-motu-jvm-port-jmx=9010
```

```
tomcat-motu-jvm-address-debug=9090
```

```
tomcat-motu-jvm-umask=tomcat|umask|0000 (More details...)
```

##### **Tomcat umask**

By default, if tomcat-motu-jvm-umask is not set, motu sets the umask with result of the command umask

```
tomcat-motu-jvm-umask=umask|tomcat|0000
```

- umask: By default, if tomcat-motu-jvm-umask is not set, motu sets the umask with result of the command umask
- tomcat: Apache Tomcat process forces umask to 0027 (<https://tomcat.apache.org/tomcat-8.5-doc/security-howto.html>)
- 0000: Custom umask value

Values 0002 or umask are recommended if Motu downloaded results are served by a frontal web server.

##### **Java property tds.http.sotimeout**

By default, this parameter is at "300". It represents the maximum delay in seconds for TDS to answer a MOTU request (reading timeout of the socket).

For queries involving lots of files, TDS might need more than the default 5 minutes to answer, and to avoid the error "004-27 : Error in NetcdfWriter finish", this parameter can be set to a higher value in:

```
tomcat-motu-jvm-javaOpts=-server [...] -XX:MaxPermSize=512M -Dtds.http.sotimeout=4000
```

##### **Java property tds.http.conntimeout**

By default, this parameter is at "60". It represents the maximum delay in seconds for TDS to accept a MOTU request (connection timeout on the socket).

The parameter can be customized and added in:

```
tomcat-motu-jvm-javaOpts=-server [...] -XX:MaxPermSize=512M -Dtds.http.conntimeout=100
```

#### 2.4.6.3.2.1 TOMCAT NETWORK PORTS

The parameters below are used to set the different network ports used by Apache Tomcat.

At startup, these ports are set in the file "\$installdir/motu/tomcat-motu/conf/server.xml".

But if this file already exists, it won't be replaced. So, in order to apply these parameters, remove the file "\$installdir/motu/tomcat-motu/conf/server.xml".

#### **tomcat-motu-port-http=9080**

# HTTPs is in a common way managed from a frontal Apache HTTPd server. If you really need to use it from Tomcat, you have to tune the SSL certificates and the protocols directly in the file "\$installdir/motu/tomcat-motu/conf/server.xml".

tomcat-motu-port-https=9443

tomcat-motu-port-ajp=9009

tomcat-motu-port-shutdown=9005

#### **2.4.6.3.2.2 CAS SSO SERVER**

# true or false to enable the SSO connection to a CAS server

cas-activated=false

# Cas server configuration to allow Motu to access it

# @see <https://wiki.jasig.org/display/casc/configuring+the+jasig+cas+client+for+java+in+the+web.xml>

# The start of the CAS server URL, i.e. <https://cas-cis.cls.fr/cas>

cas-server-url=<https://cas-cis.cls.fr/cas>

# The Motu HTTP server URL, for example: <http://misgw-ddo-qt.cls.fr:9080> or <http://motu.cls.fr>

# If you use a frontal HTTPd server, you have to know if its URL will be called once the user will be login on CAS server.

# In this case, set the Apache HTTPd server. The value will be [http://\\$apacheHTTPdServer/motu-web/Motu](http://$apacheHTTPdServer/motu-web/Motu) So, in Apache HTTPd, you have to redirect this URL to the Motu Web server

cas-auth-serverName=[http://\\$motuServerIp:\\$motuServerPort](http://$motuServerIp:$motuServerPort)

# The proxy callback HTTPs URL of the Motu server (\$motuServerIp is either the Motu host or the frontal Apache HTTPs host ip or name. \$motuServerHttpsPort is optional if default HTTPs port 443 is used, otherwise it is the same value as defined above with the key "tomcat-motu-port-https", or it is the port defined for the HTTPs server on the frontal Apache HTTPd)

cas-validationFilter-proxyCallbackUrl=[https://\\$motuServerIp:\\$motuServerHttpsPort/motu-web/proxyCallback](https://$motuServerIp:$motuServerHttpsPort/motu-web/proxyCallback)

**IMPORTANT:** Motu uses a Java HTTPs client to communicate with the CAS server. When the CAS server has an untrusted SSL certificate, you have to add it to Java default certificates or to add the Java property named "javax.net.ssl.trustStore" to target a CA keystore which contains the CAS Server SSL CA public key. For example, add this property by setting Java option tomcat-motu-jvm-javaOpts:

```
tomcat-motu-jvm-javaOpts=-server -Xmx4096M -Xms512M -XX:MetaspaceSize=128M -
XX:MaxMetaspaceSize=512M -Djavax.net.ssl.trustStore=/opt/cmems-
cis/motu/config/security/cacerts-with-cas-qt-ca.jks
```

The following part is not relevant in the CMEMS context, as the SSO CAS server has been signed by a known certification authority.

If you need to run tests with your own SSO CAS server, without any certificate signed by a known certification authority, you have to follow the following steps.

How to build the file `cacerts-with-cas-qt-ca.jks` on Motu server?

- Download the certificate file (for example "ca.crt") of the authority which has signed the CAS SSO certificate on the CAS server machine (`/opt/atoll/ssl/ca.crt`) and copy it to `"${MOTU_HOME}/config/security/"`, then rename it "cas-qt-ca.crt"
- Copy the default Java cacerts `"/opt/cmems-cis-validation/motu/products/jdk1.7.0_79/jre/lib/security/cacerts"` file into `"${MOTU_HOME}/config/security/"` and rename this file to "cacerts-with-cas-qt-ca.jks"

```
cp /opt/cmems-cis-validation/motu/products/jdk1.7.0_79/jre/lib/security/cacerts
/opt/cmems-cis/motu/config/security/
mv /opt/cmems-cis/motu/config/security/cacerts /opt/cmems-
cis/motu/config/security/cacerts-with-cas-qt-ca.jks
```

- Then import "cas-qt-ca.crt" inside "cacerts-with-cas-qt-ca.jks", Trust the certificate=yes

```
/opt/cmems-cis-validation/motu/products/jdk1.7.0_79/bin/keytool -import -v -trustcacerts -
alias $CAS_HOST_NAME -file cas-qt-ca.crt -keystore cacerts-with-cas-qt-ca.jks -keypass XXX
```

### NetCdf standard names

When NetCdf variables are read in data files, either by Threads or directly by Motu, Motu wait for a standard name metadata sttribute to be found for each variable as required by the CF convention. Due to any production constraints, some netcdf files does not have any `standard_name` attribute.

In the case, you can add directly in the configuration folder, a file named `standardNames.xml` in order to map a `standard_name` to a netcdf variable name.

You can find an example in Motu source: `/motu-web/src/main/resources/standardNames.xml`

## Supervision

To enable the status supervision, set the parameter below:

tomcat-motu-urlrewrite-statusEnabledOnHosts=localhost,\*.cls.fr

This parameter is used to set the property below in the WEB.XML file:

```
<!-- Documentation from http://tuckey.org/urlrewrite/manual/3.0/
you may want to allow more hosts to look at the status page
statusEnabledOnHosts is a comma delimited list of hosts, * can
be used as a wildcard (defaults to "localhost, local, 127.0.0.1") -->
<init-param>
  <param-name>statusEnabledOnHosts</param-name>
  <param-value>${tomcat-motu-urlrewrite-statusEnabledOnHosts}</param-value>
</init-param>
```

For more details read:

org.tuckey.UrlRewriteFilter FILTERS : see <http://tuckey.org/urlrewrite/manual/3.0/>

### 2.4.6.4 Log settings

Log are configured by using log4j2 in file config/log4j.xml.

#### 2.4.6.4.1 MOTU QUEUE SERVER LOGS: MOTUQSLOG.XML, MOTUQSLOG.CSV

This log files are used to compute statistics about Motu server usage.

Two formats are managed by this log, either XML or CSV.

To configure it, edit config/log4j.xml

Log format: XML or CSV

Update the fileFormat attribute of the node "MotuCustomLayout": A string either "xml" or "csv" to select the format in which log message are written.

Also update the log file name extension of the attributes "fileName" and "filePattern" in order to get a coherent content in relationship with value set for MotuCustomLayout file format.

If this attribute is not set, the default format is "xml".



```
<RollingFile name="log-file-infos.queue"
  fileName="${sys:motu-log-dir}/motuQSlog.xml"
  filePattern="${sys:motu-log-dir}/motuQSlog.xml.%d{MM-yyyy}"
  append="true">
  <!-- fileFormat=xml or csv -->
  <MotuCustomLayout fileFormat="xml" />
  <Policies>
    <TimeBasedTriggeringPolicy interval="1" modulate="true"/>
  </Policies>
</RollingFile>
```

### Log path

In the dissemination unit, Motu shares its log files with a central server.

Log files have to be saved on a public access folder.

Set absolute path in "fileName" and "filePattern" attributes. This path will be served by the frontal Apache HTTPd or Apache Tomcat.

For example, if you want to share account transaction log files, you edit config/log4j.xml. Update content below:

```
<RollingFile name="log-file-infos.queue" fileName="${sys:motu-log-dir}/motuQSlog.xml"
  filePattern="${sys:motu-log-dir}/motuQSlog.xml.%d{MM-yyyy}"
```

with:

```
<RollingFile          name="log-file-infos.queue"          fileName="/opt/cmems-
cis/motu/data/public/transaction/motuQSlog.xml"
  filePattern="/opt/cmems-cis/motu/data/public/transaction/motuQSlog.xml.%d{MM-yyyy}"
```

Note that both attributes fileName and filePattern have been updated.

Then the frontal Apache HTTPd server has to serve this folder.

### 2.4.6.5 Theme and Style

In Motu you can update the theme of the website. There are 2 main issues in order to understand how it works?

- [Template] velocity: The velocity templates are used to generate HTML pages from Java objects.
- [Style] CSS, Images and JS: These files are used to control style and behaviour of the web UI.

By default, the template and style are integrated in the "war". But the Motu design enables to customize it easily.

- [Template] velocity: You can change all templates defined in: `motu/tomcat-motu/webapps/motu-web/WEB-INF/lib/motu-web-2.6.00-SNAPSHOT.jar/velocityTemplates/*.vm` by defining them in `motu/config/velocityTemplates`.

The main HTML web page structure is defined by the `index.vm` velocity template. For example, if you create a file `motu/config/velocityTemplates/index.vm` containing an empty HTML page, the website will render empty web pages.

"`index.vm`" is the default theme. The name can be updated for each `motuConfig#configService` by setting `velocityTemplatePrefix=""`. By default `velocityTemplatePrefix="index"`.

- [Style] CSS, Images and JS: Those files are integrated with the default theme `motu-web-2.6.00-SNAPSHOT.war/css/`, `motu-web-2.6.00-SNAPSHOT.war/js/`. These files can be downloaded from an external server which enables to benefit to several mMotu servers at the same time. The external server's name can be updated for each `motuConfig#configService` by setting `httpBaseRef=""`.
- By default, `httpBaseRef` searches static files from the Motu web server, for example:

```
service.getHttpBaseRef()/css/motu/screen/images/favicon.ico"
```

## 2.5 Installation of the Product Factory

### 2.5.1 Development Docker

#### 2.5.1.1 Preparation of the environment

The following steps are to be carried out only once.

- Docker installation according to your environment:

```
$ sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common
$ curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb_release -cs) stable"
sudo apt update
```

```
$ sudo apt install docker-ce  
$ sudo usermod -aG docker $USER  
$ sudo reboot
```

- Connection to the docker registry

```
docker login registry.brest.cls.fr
```

- Recovery and installation of the main project

```
# here you can put the name of your choice  
ENV_NAME=metoc-wms  
cd ~/pyve  
$ git clone git@devs-brest.cls.fr:fish_WMS/main.git ${ENV_NAME}  
$ cd ${ENV_NAME}  
./bin/install_env
```

- Building the local image The following command will deploy the sources in a local image:

```
create_docker_dev.sh
```

- Initialization of the postgres database (coastline management) and creation of a volume for the persistence of the database:

```
$ docker volume create pg_db_data
```

- Create a link to the docker-compose.yml file corresponding to the configuration of the machine:

```
cd ~/pyve/metoc-wms  
ln -s config/cls/dev/docker-compose.yml
```

- Initialization of the database: ATTENTION: remember to start the docker

```
dev pull  
dev up -d  
# initialization of the gshhs database  
dev exec postgres bash  
postgres@xxxxxx:~ $ psql -c "CREATE ROLE gshhs WITH SUPERUSER LOGIN PASSWORD 'gshhs';"  
postgres@xxxxxx:~ $ createdb -O gshhs GSHHS
```

```
postgres@xxxxxx:~ $ PGUSER=gshhs PGPASSWORD=gshhs PGHOST=127.0.0.1 psql -d GSHHS -c "create
extension postgis;"

postgres@xxxxxx:~ $ exit
```

- Ingestion of coastlines:

```
$ cd virtualenv

$ xz -d --keep share/dump_gshhs.dump.xz

touch share/dump_gshhs.dump.lst

$ chmod 777 share/dump_gshhs.dump.lst

dev exec wms bash

user@xxxxxxx:~ $ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-
key add -

user@xxxxxxx:~ $ sudo apt-get install lsb-release

user@xxxxxxx:~ $ echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main"
|sudo tee /etc/apt/sources.list.d/pgdg.list

user@xxxxxxx:~ $ sudo apt-get update

user@xxxxxxx:~ $ sudo apt-get install -y postgresql-12-postgis-3 postgresql-client-12

user@xxxxxxx:~ $ perl /usr/share/postgresql/12/contrib/postgis-3.0/postgis_restore.pl
/shared/dump_gshhs.dump | PGHOST=postgis PGUSER=gshhs PGPASSWORD=gshhs psql GSHHS

user@xxxxxxx:~ $ exit
```

### 2.5.1.2 Launch of the dev service

Launch the service as follows:

```
dev pull

dev up -d
```

## 2.5.2 Production Docker

### 2.5.2.1 Preparation of the environment

- Recovery and installation of the main project
  - prerequisites:
    - git
    - docker
    - docker-compose

```
docker login registry.brest.cls.fr
$ cd ~
$ git clone http://gitlab.brest.cls.fr/fish_WMS/main.git metoc-wms
cd metoc-wms
chmod 777 cache
```

- Create a link to the docker-compose.yml file corresponding to the configuration of the machine

```
cd ~/metoc-wms
ln -s config/cls/prod/docker-compose.yml
```

- Initialization of the postgis database (coastline management)

- Creating the volume for database persistence:

```
docker volume create pg_db_data
```

- Initialization of the database: ATTENTION: remember to start the docker

```
$ docker-compose exec postgis bash
postgres@xxxxxx:~ $ psql -c "CREATE ROLE gshhs WITH SUPERUSER LOGIN PASSWORD
'gshhs';"
postgres@xxxxxx:~ $ createdb -O gshhs GSHHS
postgres@xxxxxx:~ $ PGUSER=gshhs PGPASSWORD=gshhs PGHOST=127.0.0.1 psql -d
GSHHS -c "create extension postgis;"
postgres@xxxxxx:~ $ exit
```

- Ingestion of coastlines :

```
$ xz -d --keep share/dump_gshhs.dump.xz
touch share/dump_gshhs.dump.lst
$ chmod 777 share/dump_gshhs.dump.lst
$ docker run -ti --rm --link metocwms_postgis_1:postgis -v $(pwd):/here --
network=metocwms_default registry.brest.cls.fr/common/docker_posgres_postgis:2.0
bash
user@xxxxxx:~ $ perl /usr/share/postgresql/12/contrib/postgis-3.0/postgis_restore.pl
/here/share/dump_gshhs.dump | PGHOST=postgis PGUSER=gshhs PGPASSWORD=gshhs
psql GSHHS
```

```
user@xxxxxxx:~ $ exit
```

### 2.5.2.2 Launch of the prod

If the preparation step has already been done (be careful to check that step 2 of the preparation has been done):

```
docker-compose down # stop service and clean up containers
$ git pull
$ docker-compose up -d --scale wms=4 # restart the service with 4 instances of wms
```

The `--scale` option allows to define the number of instances of the wms service.

**IMPORTANT:** do not forget to update the nginx conf with the necessary number of upstream.

Example for 4 :

```
cat config_prod/default.conf
upstream WMS {
    server metocwms_wms_1:8081;
    server metocwms_wms_2:8081;
    server metocwms_wms_3:8081;
    server metocwms_wms_4:8081;
}
```

### 2.5.3 Performance of Nginx and uwsgi

In order to improve the performance of the server, the following parameters can be used:

- **nginx.conf:**
  - *worker\_processes*: maximum number of processes (currently set to 1)
  - *worker\_connections*: maximum number of simultaneous connections that can be opened by a process (set to 10000)
  - *keepalive\_timeout*: time limit after which the request returns a timeout (set to 2 minutes)
- **default.conf:**
  - *uwsgi\_read\_timeout*: time limit to read the uwsgi server response (set to 2 minutes)
- options of the **uwsgi** command:
  - *max-requests* (-R): maximum number of requests allowed before process restart (set to 10000)
  - *min-worker-lifetime*: time limit after which the server is paused after a period of intensive use (set to 30 minutes)

- *socket-timeout* (-z): response time limit (set to 2 minutes)
- *listen* (-l): simultaneous sockets limit (set to 4096)

## 2.5.4 Nginx and uwsgi log formatting

- Nginx log format: can be configured using the Nginx configuration files:

- nginx.conf:

```
log_format main '[PID=$pid $request_id] $remote_addr - $remote_user [$time_local]
$request_method "$request_uri" ' '=> generated $body_bytes_sent bytes in
$request_time seconds ' '=> $server_protocol $status "$http_user_agent";
```

- default.conf: indicate the key corresponding to the format:

```
access_log /proc/self/fd/1 main;
```

- uwsgi logs format (WMS logs):

- default.conf: declare uwsgi\_param variables for each Nginx information we want to include in the WMS logs (e.g., the request ID).

```
uwsgi_param HTTP_X_REQUEST_ID $request_id;
```

- Then, in the uwsgi startup command (**docker-compose.yml** and/or **Dockerfile**), add the --logformat option:

```
/home/user/pyve/metocWMS/bin/uwsgi --socket :8081 --chmod-socket=666 --paste
config:/config/deployment_ogc.ini

--home /home/user/pyve/metocWMS/ -p 4 -R 10000 --min-worker-lifetime 1800 -b
32768 -z 120 -l 4096 \

--log-date='%Y:%m:%d %H:%M:%S' --logformat-strftime

--logformat '[PID=%(pid) %(var.HTTP_X_REQUEST_ID)] %(addr) - %(user)
[%{ltime}] %(method) "%(uri)" => generated %(rsize) bytes in %(secs) seconds
=> %(proto) %(status) "%(uagent)"]'
```

## 3 Operator incident procedure

### 3.1 Nagios - Ingestion

#### Important information

In order to run this procedure, you will need to have set up the Odyssea context first. Check that the Nagios check\_export\_folder is not set to CRITICAL as the error may be due to an incident upstream in the processing chain.

If check\_export\_folder is set to CRITICAL, follow this procedure Nagios - Export Folder

- Purpose of the procedure

Netcdf files are saved after ingestion in the DB in the Processed folder. The application is supposed to receive and ingest data daily. If there are no files that have been created since less than 12 hours in the Processed folder then there is a problem with the DB integration.

- Impact and backup solution

Data is no longer ingested into the DB

- Procedure

#### Prerequisites

Nagios **CRITICAL** alert for at least one hour

#### Steps

- Please connect to the computer where the **kubectl** client is installed and activate the correct context

- Check that you have activated the right context

```
root@xxxxxxx: kubectl config current-context  
k8s-prod1
```

- Get the nrpe pod id and connect to the nrpe pod (in the example below the nrpe pod id is nrpe-cf4d866f9-kjdzp)

```
root@xxxxxxx: kubectl get pods -n odyssea  
  
NAME READY STATUS RESTARTS AGE  
catalog-engine-cronjob-1611136800-t6dkt 0/1 Completed 0 48m  
catalog-engine-cronjob-1611139500-fjd2p 0/1 Completed 0 3m57s  
catalog-engine-f9c8f645c-zcdr8 2/2 Running 0 6d1h  
data-collection-5d9445dd49-gs8fl 1/1 Running 0 7d
```



```
data-collection-hangfire-7bffc9678c-d6nfq 2/2 Running 0 7d
database-dump-cronjob-1611133200-dd7xz 0/1 Completed 0 108m
frontend-back-d957f79b4-6xpl8 2/2 Running 0 9d
frontend-odyssea-76df6b8499-6b4fg 1/1 Running 0 9d
geonetwork-6d5b54c96b-v5g2v 1/1 Running 0 8d
ingestion-volume-checker-7d78c5bb48-cwcxm 1/1 Running 1 64m
nginxfrontal-786c9f5554-fs4lh 1/1 Running 0 8d
nrpe-cf4d866f9-kjdz 1/1 Running 0 8d
```

- Connect to the pod

```
root@xxxxxxx: kubectl exec -it nrpe-XXXXXXX bash -n odyssea
```

- Check if there are any files created in the last 12 hours in the Processed folder

```
root@nrpe-76f799cf78-8ft5n: /# cd /opt/Processed
root@nrpe-76f799cf78-8ft5n: /# -lrt
```

- Note: If there are files created in the last 12 hours, run the nagios check again and it should turn green. Open an incident ticket if there are not

- Check if there are any files less than 12 hours old in the Processed\_with\_error folder

```
root@nrpe-76f799cf78-8ft5n: cd /opt/Processed_with_ERROR/
root@nrpe-76f799cf78-8ft5n: ls -lrt
```

- **Case 1:** There are present files created in the last 12hrs: The application tried to push the NetCDF files in DB but there were errors. Create a SIO specifying that the Processed\_with\_error folder contains files created less than 12 hours ago and add the error log lines
- **Case 2:** No files created in the last 12 hours, open a SIO and specify that the Processed\_with\_error folder does not contain any files created in the last 12 hours and add the lines of the log in ERROR.

```
root@nrpe-76f799cf78-8ft5n: tail -f /opt/Edisoft/logs/SOSUploader-logger.log -n 500
```

#### Case encountered:

If the following message appears in the logs, **[ERROR] 2020-03-13@04:00 [main] dbBlacklist.DbBlacklist - File corrupted while reading record: null. Possible solution: use the recovery tool [90030]** this means that the Base H2 used by SOS server to list the files inserted in BDD is corrupted. You can add this information to the ticket so that IE can remove it using the procedure detailed in IE Incident Procedures

## 3.2 Nagios - Export Folder

### Important information

To carry out this procedure you will need to have set up the Odyssea context.

After retrieving files via CMEMS FTP and via HTTP, the NetCDF files are saved in the Export folder before a cron job pushes them to the DB. This cronjob normally runs every 45 minutes.

This error is reported by Nagios when the Export folder contains NetCDF files that have been created more than 2 hours ago.

- **Purpose of the procedure**

Check if the Export folder contains files that arrived in the last 2 hours and restart the cronjob

- **Impact and backup solution**

As long as the alert is present, the Net CDF files are no longer ingested into the DB

- **Procedure**

### Prerequisites

The Nagios alert has been present for over an hour and the message "Files are no longer being ingested" is returned in the nagios check status

### Steps

- Check that the Export folder contains files longer than 2 hours

- Check that you have activated the right context

```
root@xxxxxxx: kubectl config current-context  
k8s-prod1
```

- Get the nrpe pod id and connect to the nrpe pod ( in the example below the nrpe pod id is nrpe-cf4d866f9-kjdzp)

```
root@xxxxxxx: kubectl get pods -n odyssea  
  
NAME READY STATUS RESTARTS AGE  
catalog-engine-cronjob-1611136800-t6dkt 0/1 Completed 0 48m  
catalog-engine-cronjob-1611139500-fjd2p 0/1 Completed 0 3m57s  
catalog-engine-f9c8f645c-zcdr8 2/2 Running 0 6d1h  
data-collection-5d9445dd49-gs8fl 1/1 Running 0 7d  
data-collection-hangfire-7bffc9678c-d6nfq 2/2 Running 0 7d  
database-dump-cronjob-1611133200-dd7xz 0/1 Completed 0 108m
```

```
frontend-back-d957f79b4-6expl8 2/2 Running 0 9d
frontend-odyssea-76df6b8499-6b4fg 1/1 Running 0 9d
geonetwork-6d5b54c96b-v5g2v 1/1 Running 0 8d
ingestion-volume-checker-7d78c5bb48-cwcxm 1/1 Running 1 64m
nginxfrontal-786c9f5554-fs4lh 1/1 Running 0 8d
nrpe-cf4d866f9-kjdz 1/1 Running 0 8d
```

- Connect to the pod

```
root@xxxxxxx: kubectl exec -it nrpe-XXXXXXX bash -n odyssea
```

- Go to the Export folder

```
root@nrpe-76f799cf78-8ft5n:cd /Export/
```

- **Case 1:** All the files were created in the last 2 hours, restart the nagios check and open a SIO if it does not turn green
- **Case 2:** There are files created since more than 2 hours, then the ingestion is well stopped
  - We are going to check if we are not in the case where the H2 base is corrupted for that while being always connected to the pod nrpe to go and check the logs

```
root@nrpe-76f799cf78-8ft5n: cat /opt/Edisoft/logs/SOSUploader-logger.log
```

- Check if the following message is present in the logs: ***DbBlacklist - File corrupted while reading record: null. Possible solution: use the recovery tool [90030-199]***
  - The error message is present. Open an SIO so that IE deletes the H2 database (IE BDD H2 procedure - Corrupted)
  - No error message, go to next step
- Check if there is a cronjob running for more than 1 hour
  - Log into the Rancher interface <https://rancher-p1.clouds.cls.fr/> using the user clsops (Accounts and passwords Rancher interface section)
  - Go to the workload (pods) section, you can use the "Introduction to Rancher" page
  - Click on catalog-engine-cronjob and check if there is a cronjob with the status running and that has been created for more than an hour

Workload: catalog-engine-cronjob

Namespace: odyssea	Image: registry-ext.csl.fr:443/odyssea/sos-server/ingestion2.110-1	Workload Type: Cron Job
Cron Schedule: */45 * * * * Every 0 and 45th minute past every hour	Endpoints: n/a	Created: 8:40 AM Pod Restarts: 0

State	Name	Image	Node
Succeeded	test1-5h1q	registry-ext.csl.fr:443/odyssea/sos-server/ingestion2.110-1 10/12/27/23 / Created an hour ago / Restarts: 0	
Succeeded	catalog-engine-cronjob-1611136800-t6dkt	registry-ext.csl.fr:443/odyssea/sos-server/ingestion2.110-1 10/12/27/23 / Created 36 minutes ago / Restarts: 0	
Succeeded	catalog-engine-cronjob-1611135900-q57h7	registry-ext.csl.fr:443/odyssea/sos-server/ingestion2.110-1 10/12/27/23 / Created an hour ago / Restarts: 0	

- There is a job that seems to be running in a vacuum, open an SIO and notify the IE so that it applies the procedure related to a blocked ingest cronjob

- No job running for more than 1 hour go to the next step

- Try to run a cron job to try to force the ingestion process

- Exit the pod nrpe

```
root@nrpe-76f799cf78-8ft5n: exit
```

- Create a cron job with the command described below

```
root@xxxxxxx: kubectl create job --from=cronjob/catalog-engine-cronjob
<name-what-you-want> -n odyssea
```

- Connect again to the nrpe pod and check that the Export directory is empty
- If ok, restart the nagios check it should turn green, but if the case occurs again in the 2h following the procedure open a SIO
- If nok, open an SIO

### 3.3 Nagios - FTP CMEMS

#### Important information

An error is reported by Nagios when:

- The swagger services that retrieve job ids/descriptions are not accessible
- The CMEMS ftp is not accessible or the folder from which the data is retrieved has not been created for the day in question.

- **Purpose of the procedure**

Check that swagger services are functional

Check that the CMEMS FTP is accessible and/or that the folder of the day has been created

- **Procedure**

## Prerequisites

Nagios **CRITICAL** alert for at least one hour

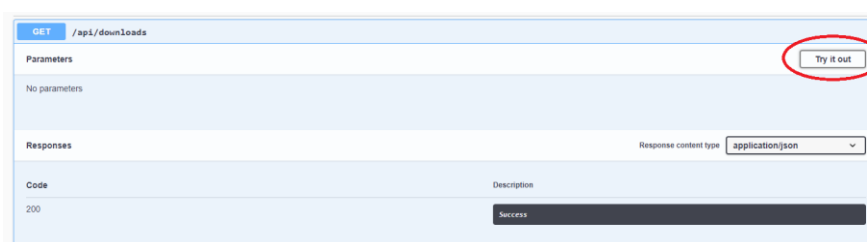
## Steps

**Case 1:** The error message in the output of the nagios check specifies the following error "the job description could not be retrieved" or "the job ID list could not be retrieved".

- Verify that the swagger is accessible by logging in using the credentials available on the Accounts and Passwords page in the **"Datacollection Component Management Interface> Swagger"** section and test the following services:

- **Test the /api/downloads service (no parameters required):**

- Click on "Try it out".



GET /api/downloads

Parameters

No parameters

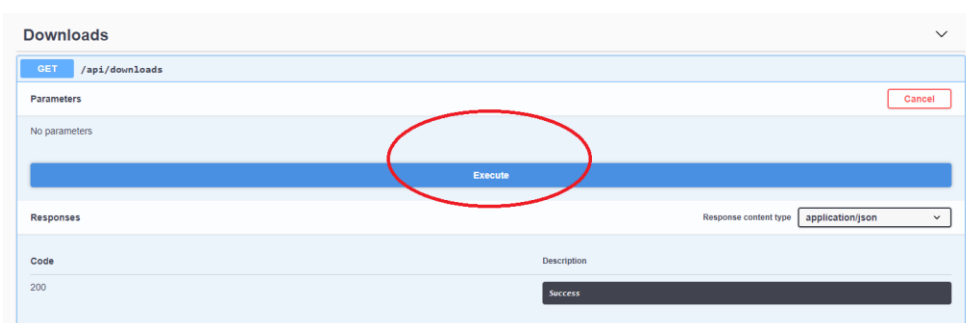
Responses

Response content type: application/json

Code: 200

Description: Success

- Click on "Execute".



Downloads

GET /api/downloads

Parameters

No parameters

Execute

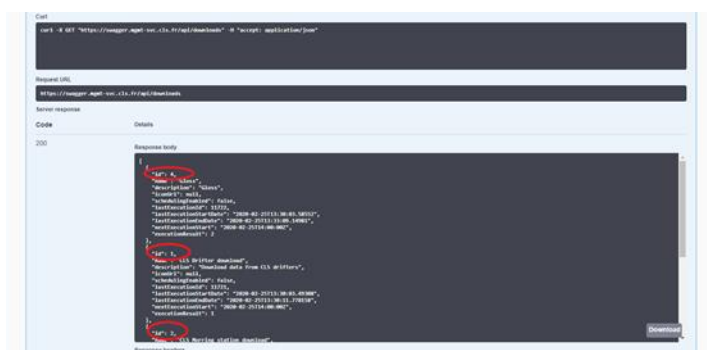
Responses

Response content type: application/json

Code: 200

Description: Success

- Normally you should have a list of jobs as return, get one of the ids back and use it for the next step which will test the **/api/downloads/json/{id}** service



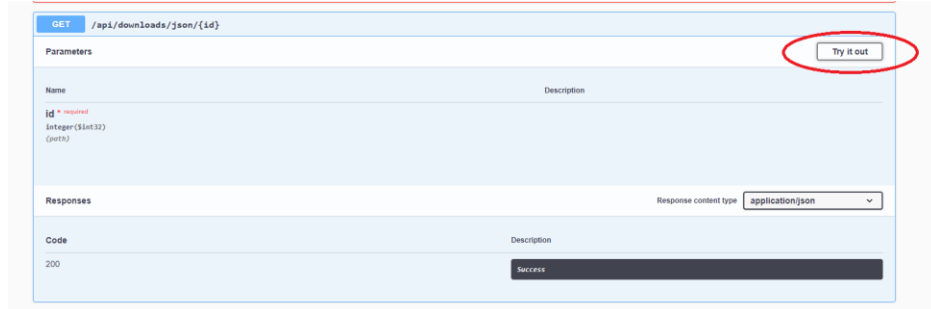
```
curl -X GET "https://swagger-api.vic.ac.uk/api/downloads" -H "accept: application/json"
```

```
Response (200)
https://swagger-api.vic.ac.uk/api/downloads
```

```
Server response
Code: 200
Details
Response body
{
  "jobs": [
    {
      "id": "1",
      "description": "Job 1",
      "status": "Success",
      "created": "2013-01-01T00:00:00Z",
      "updated": "2013-01-01T00:00:00Z",
      "deleted": "2013-01-01T00:00:00Z",
      "owner": "vic.ac.uk"
    },
    {
      "id": "2",
      "description": "Job 2",
      "status": "Success",
      "created": "2013-01-01T00:00:00Z",
      "updated": "2013-01-01T00:00:00Z",
      "deleted": "2013-01-01T00:00:00Z",
      "owner": "vic.ac.uk"
    },
    {
      "id": "3",
      "description": "Job 3",
      "status": "Success",
      "created": "2013-01-01T00:00:00Z",
      "updated": "2013-01-01T00:00:00Z",
      "deleted": "2013-01-01T00:00:00Z",
      "owner": "vic.ac.uk"
    },
    {
      "id": "4",
      "description": "Job 4",
      "status": "Success",
      "created": "2013-01-01T00:00:00Z",
      "updated": "2013-01-01T00:00:00Z",
      "deleted": "2013-01-01T00:00:00Z",
      "owner": "vic.ac.uk"
    },
    {
      "id": "5",
      "description": "Job 5",
      "status": "Success",
      "created": "2013-01-01T00:00:00Z",
      "updated": "2013-01-01T00:00:00Z",
      "deleted": "2013-01-01T00:00:00Z",
      "owner": "vic.ac.uk"
    }
  ]
}
```

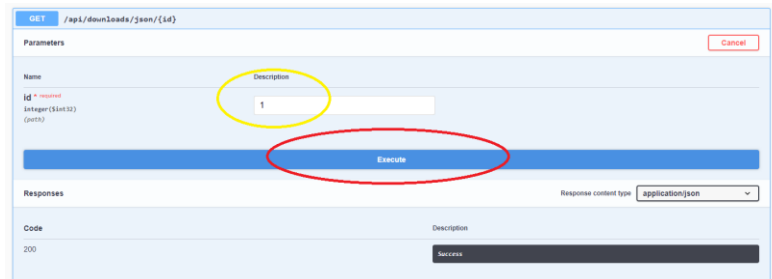
- Test the `/api/downloads/json/{id}` service which allows the retrieval of the job description ; The id parameter is to be provided and it is to be retrieved from the `/api/downloads` service response (choose one among those returned by the service)

➤ Click on "Try it out".



The screenshot shows the Swagger UI for the endpoint `GET /api/downloads/json/{id}`. The 'Parameters' section shows a required integer parameter `id` (path). The 'Responses' section shows a 200 status code with a 'success' message. The 'Try it out' button is circled in red.

➤ Fill in ID and click on Execute



The screenshot shows the Swagger UI for the endpoint `GET /api/downloads/json/{id}`. The 'id' field is filled with the value '1'. The 'Execute' button is circled in red.

- If the site is inaccessible or the service(s) do not respond open a SIO
- If the site is accessible, open a SIO for the operations engineer

**Case 2:** The error message in the output of the nagios check shows the following error "NOK the CMEMS ftp is inaccessible, or the current folder does not exist.

- Check that the CMEMS ftp is accessible by connecting to the ftp from the operators' workstations using the login available on the Accounts and Passwords page in the "FTP" section
  - If the ftp is not accessible, open a SIO
  - If the ftp is accessible go to the next step
- Go to [/Core/INSITU\\_MED\\_NRT\\_OBSERVATIONS\\_013\\_035/med\\_multiparameter\\_nrt/latest/](#)
- Check that the file with the current date in YYYYMMDD format does not exist
  - If it exists, run the check again. Open a SIO if the status has not changed
  - If it does not exist, open a SIO so that the i.e. can contact CMEMS if necessary (see special case)

### Special cases/exceptions

File of the day does not exist: Since CMEMS only keeps 30 days of data, it has happened that the file of the day is only available at the end of the day. It is advisable to wait a while and contact them if necessary.

### 3.4 Nagios - HTTP check

#### Important information

One of Odyssea's interfaces does not seem to be accessible anymore.

- Purpose of the procedure

Verify that the site in question is down and create an OIS for the operations engineer to investigate

- Impact and backup solution

If the alert is present on:

- Odyssea Front End: Users will no longer have access to the Marinomica web interface
- SOS Server: No new data will be pushed to the DB
- GeoNetwork: Mapped data can no longer be displayed
- DataCollectio : No new data will be retrieved from FTP CMEMS by HTTP GLOSS
- Static data: The service allowing to expose static data is not accessible anymore (csv file, tiff..)

- Procedure

#### Prerequisites

Nagios **CRITICAL** alert for at least 15 minutes.

#### Steps

- Check that the site in question is no longer accessible (link available in the Accounts and passwords section) Special cases/Exceptions

Name check nagios	Impacted site	Associated POD
ODYSSEA_CE_SOS_Server	SOS Server	<b>catalog-engine</b> (Docker postgres or tomcat-sos)
ODYSSEA_CE_Geonetwork	GeoNetwork	<b>geonetwork</b> (Docker tomcat-geonetwork)
ODYSSEA_FE_Odyssea	FrontEnd	<b>frontend-odyssea</b> (Docker web)
ODYSSEA_DC_Jobs	DataCollection	<b>data-collection</b> (Docker data-collection-ui)
ODYSSEA_STATIC_DATA	Service static data	<b>static-data</b> (Docker web)

- If the site is accessible run the nagios check again it should turn green. If this happens again, open an SIO for the operations engineer
- If the site is not accessible, create a SIO for the operations engineer

### Special cases/exceptions:

- **CRITICAL CASE - Socket timeout on the SOS SERVER and the Front end:**
  - The following **Socket timeout** message appears in the check return
  - Log in to the Rancher interface <https://rancher-p1.clouds.cls.fr/> using the operators' user
  - Go to the workload (pods) section, you can help yourself to the "Introduction to Rancher" page
  - Identify the problem pod (see table above)
  - Refer to the "Viewing PODS logs" section to retrieve the container logs and add them to the OIS
  - You can also add info related to the metrics (example below for the front end pod)



- **Notify the IE to decide whether to restart the pod**

## 3.5 Nagios - Check Pods

### Important information

To be able to carry out this procedure, you must first have set up the Odyssea context.

This error is reported by nagios in case a pod of the Odyssea cluster is not working anymore.

- **Purpose of the procedure**

The purpose of this procedure is to identify which pod is no longer functional.

- **Impact and backup solution**

As long as the alert is present, it may cause the application to stop.



- **Procedure**

**Prerequisites**

The Nagios alert has been present for more than 15 minutes.

**Steps**

- Please connect to the computer where the kubectl client is installed and activate the correct context.
- Check that you have activated the right context.

```
root@xxxxxxx: kubectl config current-context  
k8s-prod1
```

- Run the following command to find out which pods are stopped.

```
root@xxxxxxx: kubectl get pods -n odyssea
```

- Analyze the return of this order
  - **Case1:** If in the return of this command, there is an item with a STATUS other than Completed or Running. Open an OIS by adding the name of the element in question.
  - **Case2:** If all elements are Completed/Running. Rerun the check if it is still in Critical open an SIO for IE.

### 3.6 Nagios - Check PVC

**Important information**

To be able to carry out this procedure, you must first have set up the Odyssea context

This error is reported by nagios in case a pvc (storage volume) of the Odyssea cluster is not working / is saturated.

- **Purpose of the procedure**

The purpose of this procedure is to identify which PVC is no longer functional.

- **Impact and backup solution**

As long as the alert is present, it may cause the application to stop working or to be unable to ingest or retrieve new data.

- **Procedure**

### Prerequisites

The Nagios alert has been present for more than 15 minutes.

### Steps

- Please connect to the computer where the kubectl client is installed and activate the correct context.
- Check that you have activated the right context.

```
root@xxxxxxx: kubectl config current-context  
k8s-prod1
```

- Run the following command to find out which pods are stopped.

```
root@xxxxxxx: kubectl get pvc -n odyssea
```

- Analyze the return of this order
  - **Case1:** If in the return of this command, there is an item with a STATUS other than Completed or Running. Open an OIS by adding the name of the element in question
  - **Case2:** If all elements are Bound. Relaunch the check if it is still in Critical open a SIO for the IE

## 3.7 Nagios - Check PVC Size

### Important information

This error is reported by nagios when a PVC (storage volume) of the Odyssea cluster has exceeded 90% of its dedicated volume

- **Purpose of the procedure**

The purpose of this procedure is to identify which pvc is no longer functional and to create a SIO for IE

- **Procedure**

### Prerequisites

The Nagios alert has been present for more than 15 minutes.

### Steps

- Open a SIO for the IE indicating that it is the saturated volume, then will see if it can make room or if it must enlarge the PVC (i.e. procedure).

Check	Saturated PVC	POD	Path
ODYSSEA_SYS_PVC_SOS	BDD SOS	catalog-engine	/var/lib/pgsql/9.6
ODYSSEA_SYS_PVC_DC_POSTGRES	BDD DataCollection	data-collection-hangfire	/var/lib/pgsql/12
ODYSSEA_SYS_PVC_DC_DWD	Datacollection Log Directory	data-collection-hangfire	/Downloads
ODYSSEA_SYS_PVC_MONGO	BDD Mongo Front End	frontend-back	/data/db
ODYSSEA_SYS_PVC_MONGO_CONF	Conf mongo	frontend-back	/db
ODYSSEA_SYS_PVC_PROCESSED	Processed folder used by SOS Server	nrpe	/opt/Processed
ODYSSEA_SYS_PVC_PROCESSED_ERROR	Processed with error folder used by SOS Server	nrpe	/opt/Processed_with_ERROR
ODYSSEA_SYS_PVC_SOS_WKDIR	Log file/application of the ingestion engine	nrpe	/opt/Edisoft
ODYSSEA_SYS_PVC_DUMP_DB	Folder with the odyssea DB dump before pusher them to S3 (folder with purge of files older than 30 days)	nrpe	/opt/odyssea/

- The operations engineer will check that there is nothing wrong and will remove files on the volume in question or expand it if necessary. Even if you remove files in the folder, you sometimes have to wait for the rotation of the snapshots to really clean up the folder (and so that the alarm can disappear), so it is better to extend the capacity of the volume in question (IE procedure)

### 3.8 Nagios – Product Factory Trix / WavePower files

#### **Important information**

A service executed from the Odyssea application allows to generate a job that will generate once a day 4 files for the Trix and WavPower netCDF data that will be deposited on the CLS FTP to be later ingested in the CLS TDS/Motu

It is possible that the check:

- HOA-DSTR\_recup-ODYSSEA-trixMedSea (for Trix data)
- HOA-DSTR\_recup-ODYSSEA-wavePower (for WavePower data)

Or also in error because it is the task that monitors the retrieval of files from the FTP CLS in order to ingest them into the TDS/MOTU 7j7.

- **Purpose of the procedure**

An error is reported by Nagios when:

- The ftp is not accessible
- The trix files have not been generated (or only partially)

- **Impact and backup solution**

As long as the alert is present, the data in the TRIX dataset is not updated.

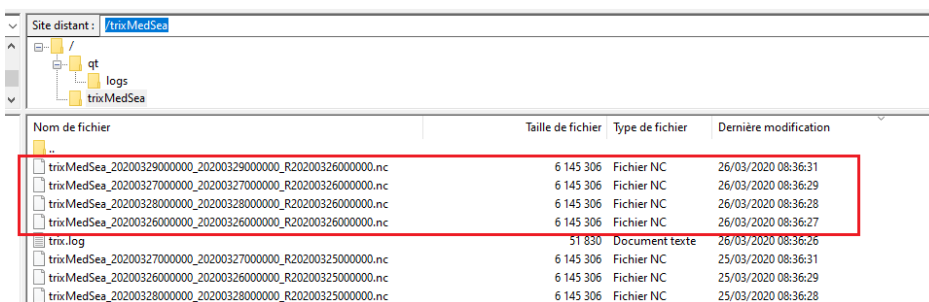
- **Procedure**

**Prerequisites**

Nagios alert ODYSSEA\_WPS\_Check\_trix\_files or ODYSSEA\_WPS\_Check\_wavepower\_files **CRITICAL** for at least one hour.

**Steps**

- Check that the ODYSSEA FTP is accessible by connecting to ftp from the operators' workstations using the identifiers available on the Accounts and passwords page in the "FTP" section.
  - If the ftp is not accessible, open a SIO.
  - If the ftp is accessible go to the next step.
- Refer to the case that concerns you
  - **Case1: ODYSSEA\_WPS\_Check\_trix\_files in CRITICAL**
    - Go to the directory: **/trixMedSea/**
    - Check if there are 4 files that have been created for the day in question  
**example:** It is the 26/03 I have 4 files that date from the 26/03



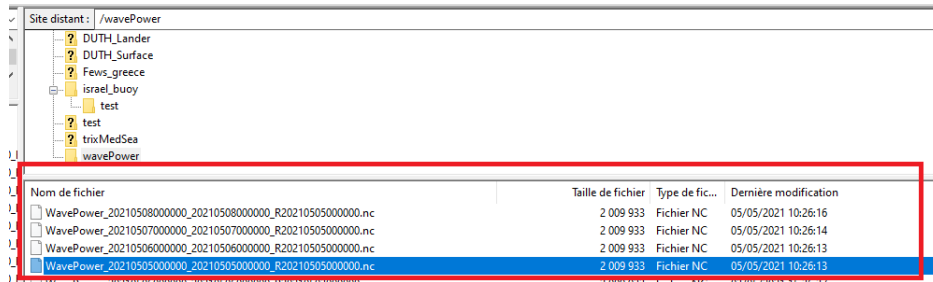
Nom de fichier	Taille de fichier	Type de fichier	Dernière modification
trixMedSea_20200329000000_20200329000000_R20200326000000.nc	6 145 306	Fichier NC	26/03/2020 08:36:31
trixMedSea_20200327000000_20200327000000_R20200326000000.nc	6 145 306	Fichier NC	26/03/2020 08:36:29
trixMedSea_20200328000000_20200328000000_R20200326000000.nc	6 145 306	Fichier NC	26/03/2020 08:36:28
trixMedSea_20200326000000_20200326000000_R20200326000000.nc	6 145 306	Fichier NC	26/03/2020 08:36:27
trix.log	51 830	Document texte	26/03/2020 08:36:26
trixMedSea_20200327000000_20200327000000_R20200325000000.nc	6 145 306	Fichier NC	25/03/2020 08:36:31
trixMedSea_20200326000000_20200326000000_R20200325000000.nc	6 145 306	Fichier NC	25/03/2020 08:36:29
trixMedSea_20200328000000_20200328000000_R20200325000000.nc	6 145 306	Fichier NC	25/03/2020 08:36:28

- If they exist, re-run the check and open a SIO if the status has not changed.

- If they don't exist, open a SIO for IE to check if the file generation job is still running.

➤ **Case2: ODYSSEA\_WP ODYSSEA\_WPS\_Check\_wavpower\_files in CRITICAL**

- Go to the directory: **/wavePower /**
- Check if there are 4 files that have been created for the day in question  
**example:** It is the 05/05 I have 4 files that date from 05/05.



Nom de fichier	Taille de fichier	Type de fichier	Dernière modification
WavePower_20210508000000_20210508000000_R20210505000000.nc	2 009 933	Fichier NC	05/05/2021 10:26:16
WavePower_20210507000000_20210507000000_R20210505000000.nc	2 009 933	Fichier NC	05/05/2021 10:26:14
WavePower_20210506000000_20210506000000_R20210505000000.nc	2 009 933	Fichier NC	05/05/2021 10:26:13
WavePower_20210505000000_20210505000000_R20210505000000.nc	2 009 933	Fichier NC	05/05/2021 10:26:13

- If they exist, re-run the check and open a SIO if the status has not changed.
- If they don't exist, open a SIO for IE to check if the file generation job is still running.

### 3.9 Nagios - Check Pod Restart

- **Purpose of the procedure**

The check indicates that one of the pods that make up the ODYSSEA application has restarted. This can be due to a CPU/RAM problem or network problem for example.

- **Impact and backup solution**

One of the pods has restarted, which may affect the proper functioning of the application.

- **Procedure**

#### **Prerequisites**

Nagios alert with the status **WARNING**

#### **Steps**

- Check the status of other Odyssea application checks
  - If checks are CRITICAL, contact IE directly to investigate
  - There is just this warning, it is less serious, but IE must investigate, you can apply the procedure. Go to the next point.
- 
- Log in to the Rancher interface <https://rancher-p1.clouds.cls.fr/> using the operator user.
- Go to the workload (pods) section, you can help yourself to the "Introduction to Rancher" page.

- Identify which pod has restarted, for this you can either
  - You refer to the output of the Nagios check, the pod name is visible
  - Look in the pod list if a pod has the restart field with a value different from 0



- Refer to the View PODS logs section to retrieve the logs of the previous crashed container.
- Adding info to a SIO for IE.

## 4 IE incident procedure

### 4.1 BDD H2 - Corrupted

#### Important information

To be able to carry out this procedure, you must first have set up the ODYSSEA context.

- Purpose of the procedure

Netcdf files are supposed to be ingested at a regular frequency into the DB. But it happened that the H2 database used by SOS Server to list the ingested files was corrupted. It is advisable (after having checked that this is the case) to delete the H2 database so that the ingestion can take place again.

- Impact and backup solution

Data is no longer ingested into the DB

- Procedure

#### Prerequisites

Nagios alerts you to the **CRITICAL** ingestion and you should check the SOS Server logs for the following message: **[ERROR] 2020-03-13@04:00 [main] dbBlacklist.DbBlacklist - File corrupted while reading record: null. Possible solution: use the recovery tool [90030]**

#### Steps

- Please connect to the computer where the kubectl client is installed and activate the correct context.
- Check that you have activated the right context.

```
root@xxxxxxx: kubectl config current-context  
k8s-prod1
```

- Connect to the nrpe pod.

```
root@xxxxxxx: kubectl exec -it nrpe-XXXXXXX bash
```

- Check the SOS Server logs to see if the error is present. To identify the file in question (there may be several per day) try to identify from the history of the nagios ingestion alert when it went to CRITICAL.

```
root@nrpe-76f799cf78-8ft5n: cat /opt/Edisoft/logs/SOSUploader-logger-XXXX.log
```

- No error message related to the H2 database, so the problem is elsewhere, there is no point in deleting the H2 database.

- The message detailed in the prerequisites is present go to the next step.
- We will now move the corrupted DB to a folder to keep it for later analysis.
- The next time the database is ingested, the H2 base will be automatically recreated.

```
root@nrpe-76f799cf78-8ft5n:cd /opt/Edisoft/ -> Move to the folder where the
H2 DB is located

root@nrpe-76f799cf78-8ft5n: mkdir saveH2BDD -> We create a folder to which
we will move the corrupted DB

root@nrpe-76f799cf78-8ft5n: mv blacklist.mv.db saveH2BDD/. -> Move the file
to this folder
```

- We must check that at the next ingestion, the files are again pushed to the DB and that the blacklist.mv.db file is well recreated

## 4.2 Extension of a PVC

### Important information

To be able to carry out this procedure, you must first have set up the Odyssea context

- **Purpose of the procedure**

It may happen that the size of the PVC is no longer sufficient and that the size of the PVC has to be extended. This procedure will show you how to do this extension.

- **Impact and backup solution**

The PVC size provisioned is no longer sufficient.

- **Procedure**

### **Prerequisites**

Nagios alert of a PVC in **WARNING / CRITICAL**

### **Steps**

- Please connect to the computer where the kubectl client is installed and activate the correct context.
- Check that you have activated the right context.

```
root@xxxxxxx: kubectl config current-context
k8s-prod1
```



- Get a pvc file of the component you are interested in <https://gitshare.cls.fr/odyssea/odyssea-deployment/-/tree/master/k8s/cls/common> :
  - frontend/frontend-back-pvc.yaml
  - catalog-engine/catalog-engine-pvc.yaml
  - datacollection/data-collection-pvc.yaml
- Make a copy of this file and modify the storage field (example below for the apilogs pvc of datacollection)

```

root@xxxxxxx: cp data-collection-pvc.yaml data-collection-pvc-modif.yaml
root@xxxxxxx: vi data-collection-pvc-modif.yaml
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: apilogs
spec:
  storageClassName: premium-multisite
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi -> Change to 10 or other (10Gi)

```

- Apply the modification (error messages on the other pvc may appear because they already exist, so do not take them into account)

```

root@xxxxxxx: kubectl apply -f data-collection-pvc-modif.yaml -n odyssea

```

- Check from the Rancher interface (Volumes section) that the capacity has changed
- or by using the kubectl command below

```

root@xxxxxxx: kubectl get pvc -n odyssea
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

```

```

apilogs Bound pvc-3e3ea977-a7b2-44ba-96d6-99658d47899c 1Gi RWO premium-multisite 140d
data-pvc Bound pvc-0e5d6ae2-f7ff-4830-bbe6-5dcf7004d9f3 50Gi RWX premium-multisite 134d
download Bound pvc-db4e9357-33b6-4ddc-9cdf-67c4c4f84f3e 5Gi RWX premium-multisite 140d
dwdlogs Bound pvc-417e7005-cbc9-4f9f-ab11-71c66db13e61 1Gi RWX premium-multisite 140d
processed-ingestion Bound pvc-52c9c87c-7758-44ec-b0e9-52bf6daa75f9 20Gi RWX premium-multisite 140d
processed-with-error-ingestion Bound pvc-53b412f3-17da-4186-8c4b-c893828095ac 20Gi RWX premium-multisite 140d
redis-pvc Bound pvc-3b025160-9df8-44f8-956e-3eda96219d95 100Mi RWO premium-multisite 134d
scripts Bound pvc-8ea6eb11-f9fa-4e23-bca1-dda79919b58f 20Gi RWX premium-multisite 133d
sos-ingestion-init Bound pvc-f28a4922-5efb-48a2-92d7-fb96a17dd202 1Gi RWO premium-multisite 140d
sos-ingestion-wkdir Bound pvc-2aec4813-4747-4ee2-856d-df89dd7652c9 21Gi RWX premium-multisite 134d
web-logs Bound pvc-6da27235-a604-40cb-a611-712cbcc85fab 3Gi RWX premium-multisite 100d

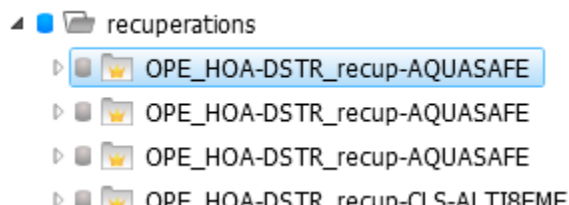
```

- Run the Nagios check again, the alert should be gone

### 4.3 Aquasafe file not produced

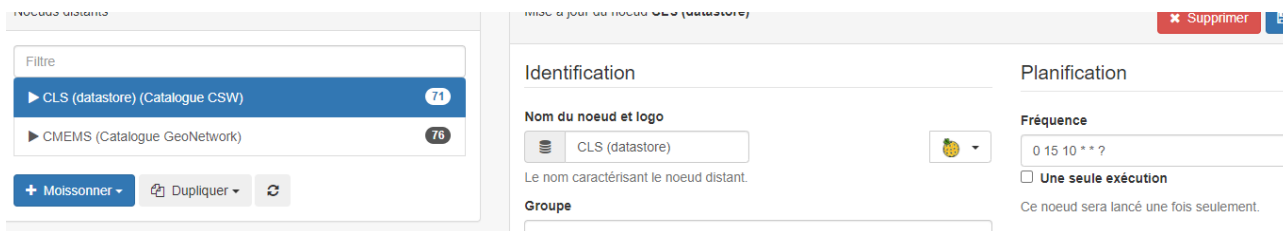
- **Purpose of the procedure**

This procedure gives you information about the Aquasafe products to solve a possible problem. Aquasafe data is not produced by the Marinomica application unlike Trix/Wavepower. These files are deposited on the ftp CLS Accounts and passwords in the directories Aquasafe\_morocco / Aquasafe\_israel / Aquasafe\_algeria by the company Hidromod. Once deposited on the FTP a ControlIM job (4 days retention) will retrieve the data (data visible under /data/atoll02/odyssea/aquasafe on the server atoll@ddo-cls.vlndata.cls.fr).



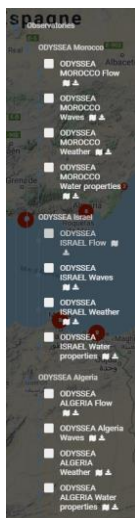
and include them in the datastore ([https://motu-datastore.cls.fr/motu\\_web/Motu?action=listcatalog&service=ODYSSEA\\_data-TDS](https://motu-datastore.cls.fr/motu_web/Motu?action=listcatalog&service=ODYSSEA_data-TDS)).

Once included in the datastore via the GeoNetwork deployed on Marinomica, we will come and harvest the data (harvesting once a day around 10H UTC).



This will allow to display the mapped data directly from the frontend.

Marinomica (odyssea)



- **Impact and backup solution**

No alarms are sent back to Nagios, because the production of this data is often irregular.

- **Procedure**

**Prerequisites**

Aquasafe Israel/Morocco/Algeria mapped data not visible from Marinomica interface.

**Steps**

- If there is a problem with the display of the Aquasafe mapped data you can check for example that the other mapped data in the datastore are displayed correctly, e.g., Derived Products/Trix(Eutrophication indices) or Wavepower.
  - If they are not displayed, the problem is not related to the Aquasafe data itself, but may be a problem with the TDS 7d/7d (check with Nagios to make sure there is not an error check on the TDS/motu).
  - If Trix is displayed, then the problem is with the Aquasafe data. Some things to check for example:

- Check on the FTP that data has been recently uploaded (if there is nothing for several days you can contact joao.rodrigues@hidromod.com in English by putting Nicolas Granier in cc).
- Check that on /data/atoll02/odyssea/aquasafe on the server atoll@ddo-cls.vlandata.cls.fr there is not a corrupted file (with a different size than the other netCDF files which could explain that the recovery is blocked).
- Possibly look at the logs from the ControlIM interface

#### 4.4 Product Factory: TRIX files not produced

- **Purpose of the procedure**

A job that runs once a day will generate TRIX files, but it can happen that the job fails to produce data. This can be due to missing files on the motu or we have a robustness defect on the motuclient, which when the connection is secured can lead in error, while the file has not finished downloading (only cases encountered for the moment).

- **Impact and backup solution**

No production of TRIX files, which can also cause an alarm in the datastore.

- **Procedure**

##### **Prerequisites**

The ODYSSEA\_WPS\_Check\_trix\_files check with a **CRITICAL** status.

##### **Steps**

- Display the list of jobs to retrieve the ID of the job you want to investigate.
- Check the logs for the job in question and verify that you are not in a special case.

```
ubuntu@xxxxxxxxxxx:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs$ ./job_log.sh
ALGO_SCHEDULER_TRIX_2020-05-27_12_50_03_076
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<      wps:Result      xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
.....
2020-05-27T12:51:33.819 INFO Job Status is Failed
```

```
2020-05-27T12:51:33.820 ERROR Execution ended with bad status 'Failed'
2020-05-27T12:51:33.821 WARN End of periodic launch of service trix of algo
trix with no result
2020-05-27T12:51:33.883 INFO Next launch in 23 hours 58 minutes 29 seconds
</wps:Data>
I'm not sure how to do that.
</wps:Result>
```

- If the following message appears in the job logs when you run the `./job_log.sh <id job>` command ( see the WPS - Script Curl page for help running this command if needed)
- This can come from:
  - Input data that is used by TRIx that is not available from CMEMS.
  - A known bug that will be solved in a future release due to a robustness defect on the clientword. Indeed, when the connection is secured (this is the case for Odyssea), it can exit while the file has not finished downloading (no error in the logs, the files are present, but the algorithm is stopped before execution).
  - Or it could be a new error, in either case you need to look at the pods logs where you will get more details than from the API request.
- We can connect to the TRIx pod to check which case we are in:
  - List the available pods to retrieve the TRIx pod ID.

```
ubuntu@xxxxxxx:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs$ kubectl get pods -n odyssea
NAME READY STATUS RESTARTS AGE
catalog-engine-77b9b5bd4-9b6d9 2/2 Running 4 29d
nginxfrontal-8f6679bcd-7g8j8 1/1 Running 3 29d
nrpe-7f64b79ffc-xx27r 1/1 Running 2 29d
redis-7888578b-65c6d 1/1 Running 2 29d
registry-d4c5f854-cw82s 1/1 Running 2 29d
scheduler-6856d9bdd8-68p7g 1/1 Running 0 21m
trix-8668845f6f-zc27l 1/1 Running 0 21m
wps-ftp-cronjob-1590408000-2qht9
```

- Two solutions are possible : look at the logs redirected to the stdout output or look at the logs available from the pod (stdout are sometimes more meaningful) :

### ❖ Solution 1: stdout

- Run the following command and analyze the logs

```
ubuntu@xxxxxx:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs$ kubectl logs trix-8668845f6f-zc27l -n
odyssea
```

### ❖ Solution 2: from pod trix

- Connect to pod trix

```
ubuntu@xxxxxx:~/odyssea/odyssea-
deployment/k8s/cls/common/deploy/wpsjobs$ kubectl exec -it trix-8668845f6f-zc27l
bash -n odyssea
```

- Go to the directory that contains the Trix algo logs

```
[runodyssea@xxxxxxxxx wps-trix]$ cat /data/odyssea/trix/<ID OF
YOURJOB>/outputs/trix.log
```

- You can start a new job if the word was not accessible or if the log does not show any errors.
- Check on the FTP that the files have been generated.
- After checking the logs, if the error is related to the code or unknown, send the information to Jérôme Doumerc [jdoumerc@groupcls.com](mailto:jdoumerc@groupcls.com)/Sylvain Marty [smarty@groupcls.com](mailto:smarty@groupcls.com) ( + copy [ngrancier@groupcls.com](mailto:ngrancier@groupcls.com)).
- **Case encountered:**
  - **File not available at CMEMS:**
    - Check if the job has stopped when downloading a file and that no error message is present. In the example below no error is present, the algorithm stopped at the download stage.
    - Check that the last file the TRIX algorithm tried to download is present and accessible on the CMEMS website
      - In the example above the last file, we tried to download is the dataset **MEDSEA\_ANALYSIS\_FORECAST\_BIO\_006\_014 med-ogs-nut-an-fc-d**
        - ❖ Go to the following link [https://resources.marine.copernicus.eu/?option=com\\_csw&task=results?option=com\\_csw&view=details&product\\_id=MEDSEA\\_ANALYSIS\\_FORECAST\\_BIO\\_006\\_014](https://resources.marine.copernicus.eu/?option=com_csw&task=results?option=com_csw&view=details&product_id=MEDSEA_ANALYSIS_FORECAST_BIO_006_014)(replace **product\_id** with the dataset that corresponds to your case)

- ❖ Check in the services tab, that all services of the med-ogs-nut-an-fc-d dataset are accessible (note: on the CMEMS site med-ogs-nut-an-fc-d corresponds to [med00-ogs-nut-an-fc-d](#))
  - If it is not accessible contact [servicedesk.cmems@mercator-ocean.eu](mailto:servicedesk.cmems@mercator-ocean.eu) (see example in <https://jira-ext.cls.fr/browse/SIO-74737>)
  - If all the data is accessible, you can try to re-run a job at the end of the day. If it doesn't work, you'll have to re-run the previous day's run (see the [WPS - ScriptCurl](#) page the Script execute\_job.sh section which indicates how to re-run a job)
- **Timeout:**
  - For the timeout a patch has been installed on 25/05 and it is enough to modify the environment variable MOTU\_TIMEOUT in the config map wps-config to increase the period fixed at 300 s for the moment.

## 4.5 Product Factory: Wave Power file not produced

- **Purpose of the procedure**

A job that runs once a day will generate WavePower files, but sometimes the job may fail to produce data. This may be due to missing files on the CMEMS motu or a timeout on the download

- **Impact and backup solution**

No WavePower files are generated, which can also cause an alarm in the datastore (HOA-DSTR\_recup-ODYSSEA-wavePower)

- **Procedure**

### **Prerequisites**

The ODYSSEA\_WPS\_Check\_wavepower\_files check with a **CRITICAL** status

### **Steps**

- Display the list of jobs to retrieve the ID of the job you want to investigate
- Check the logs for the job in question and verify that you are not in a special case
  - If the following message appears in the job logs when you run the ./job\_log.sh <id job> command ( see the WPS - Script Curl page for help running this command if needed)

```
ubuntu@xxxxxxx:~/odyssea/odyssea-deployment/k8s/cls/common/deploy/wpsjobs$
./job_log.sh ALGO_SCHEDULER_WAVE_2021-05-06_14_25_58_086
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
< wps:Result xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
2020-05-27T12:51:33.819 INFO Job Status is Failed
2020-05-27T12:51:33.820 ERROR Execution ended with bad status 'Failed'
2020-05-27T12:51:33.821 WARN End of periodic launch of service trix of algo trix with no
result
2020-05-27T12:51:33.883 INFO Next launch in 23 hours 58 minutes 29 seconds
</wps>Data>
I'm not sure how to do that.
</wps:Result>
```

- This can come from:
  - Input data used by WavePower that is not available from CMEMS.
  - A known bug that will be solved in a future release due to a robustness defect on the motu client. Indeed, when the connection is secured (this is the case for Odyssea), it can exit while the file has not finished downloading (no error in the logs, the files are present, but the algorithm is stopped before execution).
  - Or it could be a new error, in either case you need to look at the pods logs where you will get more details than from the API request.
- We can connect to the wave pod to check which case we are in:
  - List the available pods to retrieve the WavePower pod ID (for example wave-5f7d9cd5d6-psmkz in the example below)

```
root@xxxxxxxx:~# kubectl get pods -n odyssea
NAME READY STATUS RESTARTS AGE
catalog-engine-7fb5bbcf6c-z7lc5 2/2 Running 0 14d
frontend-back-764db4766-dcbbf 2/2 Running 0 34d
frontend-odyssea-964f5c5dc-kbbm2 1/1 Running 0 14d
geonetwork-77978d567c-4w5n9 1/1 Running 0 12d
nginxfrontal-685989dc79-k58jp 1/1 Running 0 12d
nrpe-79ff694cdf-8xxsz 1/1 Running 0 12d
redis-6494896c99-85rh5 1/1 Running 0 12d
registry-788c6544f6-z6q4r 1/1 Running 0 12d
scheduler-68799f7fc8-gjhrc 1/1 Running 0 12d
trix-5d78669b96-jjqwc 1/1 Running 0 12d
wave-5f7d9cd5d6-psmkz 1/1 Running 0
```



- Two solutions are possible : look at the logs redirected to the stdout output or look at the logs available from the pod (stdout are sometimes more meaningful).

❖ **Solution 1: stdout**

- Run the following command and analyze the logs

```
ubuntu@xxxxxx:~/odyssea/odyssea-  
deployment/k8s/cls/common/deploy/wpsjobs$ kubectl logs wave-5f7d9cd5d6-psmkz -n  
odyssea
```

❖ **Solution 2 from the wavepower pod**

- Connect to the pod wave

```
ubuntu@xxxxxx:~/odyssea/odyssea-  
deployment/k8s/cls/common/deploy/wpsjobs$ kubectl exec -it wave-5f7d9cd5d6-psmkz  
bash -n odyssea
```

- Go to the directory that contains the Trix algo logs

```
odyssea@wave-5f7d9cd5d6-psmkz wps-wave]$ cat /data/odyssea/wave/<ID OF  
YOURJOB>/outputs/wavepower.log
```

- You can start a new job if the word was not accessible or if the log does not show any errors.
- Check on the FTP that the files have been generated.
- After checking the logs, if the error is related to the code or unknown, send the information to Jérôme Doumerc [jdoumerc@groupcls.com](mailto:jdoumerc@groupcls.com)/Sylvain Marty [smarty@groupcls.com](mailto:smarty@groupcls.com) ( + copy [ngrancier@groupcls.com](mailto:ngrancier@groupcls.com)).

- Case encountered:**

- Timeout:**

- For the timeout a patch has been installed on 25/05 and it is enough to modify the environment variable MOTU\_TIMEOUT in the config map wps-config to increase the period fixed at 300 s for the moment.

## 4.6 Blocked ingestion

### **Important information**

To be able to carry out this procedure, you must first have set up the ODYSSEA context

- Purpose of the procedure**

Netcdf files are supposed to be ingested at a regular frequency into the DB. But it happened that the cronjob which allows to get the files in the /Export folder and to ingest them in the DB is blocked because of a file and that it runs in the vacuum. This procedure will allow to unblock this situation.

- **Impact and backup solution**

Data is no longer ingested into the DB

- **Procedure**

### **Prerequisites**

Nagios alert of CRITICAL ingestion ODYSSEA\_DC\_Check\_Export\_Folder and possibly also ODYSSEA\_CE\_Check\_Ingestion

### **Steps**

- We will check that we are in this case by checking that we have a cronjob catalog-engine-cronjob that has been running for at least 1 hour (which is not the case in the example below).

▼ Pods  
Pods in this workload

Download YAML ⬇ Delete 🗑

<input type="checkbox"/>	State ↕	Name ↕	Image ↕	Node ↕
<input type="checkbox"/>	Succeeded	catalog-engine-cronjob-1611157500-f88zj	registry-ext.cls.fr-443/odyssea/sos-server/ingestion:2.110-1 10.42.27.21 / Created 5 minutes ago / Restarts: 0	
<input type="checkbox"/>	Succeeded	catalog-engine-cronjob-1611154800-xp5x6	registry-ext.cls.fr-443/odyssea/sos-server/ingestion:2.110-1 10.42.186.55 / Created an hour ago / Restarts: 0	
<input type="checkbox"/>	Succeeded	catalog-engine-cronjob-1611153900-b6jgb	registry-ext.cls.fr-443/odyssea/sos-server/ingestion:2.110-1 10.42.186.31 / Created an hour ago / Restarts: 0	

- We will delete the catalog-engine-cronjob pod (**DO NOT DELETE THE POD catalog-engine!!!**)

Namespace: odyssea

<input type="checkbox"/>	Active	catalog-engine	registry-ext.cls.fr-443/odyssea/sos-server/postgres:9.6.3-2 • 3 images 1 Pod / Created 3 months ago / Pod Restarts: 0	1	
<input checked="" type="checkbox"/>	Active	catalog-engine-cronjob	registry-ext.cls.fr-443/odyssea/sos-server/ingestion:2.110-1 3 Pods / Created 8 hours ago / Pod Restarts: 0		Edit Clone Redeploy Add a Sidecar Suspend View/Edit YAML New in API Delete
<input type="checkbox"/>	Active	data-collection	registry-ext.cls.fr-443/odyssea/data-collection/c7 api ui:473 • 1 image 1 Pod / Created 13 days ago / Pod Restarts: 0		
<input type="checkbox"/>	Active	data-collection-hangfire	registry-ext.cls.fr-443/odyssea/data-collection/c7 db:12.475 • 3 images 1 Pod / Created 13 days ago / Pod Restarts: 0		
<input type="checkbox"/>	Active	database-dump-cronjob	registry-ext.cls.fr-443/odyssea/odyssea-deployment/odysseadump:1.5.0 1 Pod / Created 7 hours ago / Pod Restarts: 0		
<input type="checkbox"/>	Active	frontend-back	registry-ext.cls.fr-443/odyssea/frontend/mongo:v0.8.4 • 1 image 1 Pod / Created 3 months ago / Pod Restarts: 0		
<input type="checkbox"/>	Active	frontend-odyssea	registry-ext.cls.fr-443/odyssea/frontend/web:v0.8.4 1 Pod / Created 2 months ago / Pod Restarts: 0		
<input type="checkbox"/>	Active	geonetwork	registry-ext.cls.fr-443/odyssea/catalog:3.10.3-1 • 2 images 1 Pod / Created 3 months ago / Pod Restarts: 0		

- We will recreate the cronjob to check that you have activated the right context (put yourself on the machine where you have pulled the odyssea git repo).

```
root@xxxxxxx: kubectl config current-context
```

```
k8s-prod1
```

- Recreate the cronjob POD

```
root@xxxxxxx: cd /*****/k8s/cls/common/catalog-engine
root@xxxxxxx: kubectl create -f catalog-engine-cronjob-ingestion.yaml -n
odyssea
```

- Create a cronjob and check via Rancher that it finishes well (you can also check by connecting to the NRPE pod that the /Export folder is empty).

```
root@xxxxxxx: kubectl create job --from=cronjob/catalog-engine-cronjob
<name-what-you-want> -n odyssea
```

- Once the cronjob is finished, it is possible that the alarm on the /Export folder is still present because the NetCDF file that blocked the ingestion is still present in this folder. The bug has already been reported to the developers so we will remove it.

- List the pods to get the id of the pod we are interested in

```
root@xxxxxxxxxx:~# kubectl get pods -n odyssea
NAME READY STATUS RESTARTS AGE
catalog-engine-cronjob-1611154800-xp5x6 0/1 Completed 0 62m
catalog-engine-cronjob-1611157500-e88zj 0/1 Completed 0 17m
catalog-engine-cronjob-1611158400-pf72q 0/1 Completed 0 2m12s
data-collection-hangfire-7bffc9678c-d6nfq 2/2 Running 0 7d5h
database-dump-cronjob-1611133200-dd7xz 0/1 Completed 0 7h2m
```

- Connect to the data-collection-hangfire pod and go to the /Export folder

```
root@sel-mokhtari-ramus-instance1:~# kubectl exec -it data-collection-hangfire-
7bffc9678c-d6nfq -c hangfire -n odyssea bash
root@data-collection-hangfire-7bffc9678c-d6nfq: [/app]
[#:0 / j:0 / g:] cd /Export/
```

- Move the oldest file (normally more than 2 hours old) that seems to be causing a problem

```
[#:0 / j:0 / g:] cd /Export/
```

```
[#:0 / j:0 / g:] mv XXXX /tmp/
```

- Restart the check it should turn green again and monitor the intake over the next few hours

## 4.7 Ingestion Gloss blocked

- **Purpose of the procedure**

The check indicates that the ingestion of gloss type data is stopped or blocked.

Make sure that the following Nagios checks do not also have a CRITICAL status, as the error may be due to an incident upstream of the processing chain.

- **Check\_export\_folder** only is also in **CRITICAL** go directly to the following procedure Export Folder
- **Check\_ingestion** only is also in **CRITICAL** go directly to the following procedure Ingestion
- **Check\_export\_folder + check\_ingestion** are in **CRITICAL** go directly to the following procedure Export Folder

- **Impact and backup solution**

The ingestion of this type of data is stopped

- **Procedure**





### **Prerequisites**

Nagios alert with **CRITICAL** status

### **Steps**

- Connect to the Rancher interface <https://rancher-p1.clouds.cls.fr/> using the user clsops.
- Go to the workload (pods) section, you can help yourself to the "Introduction to Rancher" page.
- Restart the 2 pods of the component datacollection (Pod data-collection-hangfire then Pod data-collection). For that, click right then on redeploy (see image below).

- Identify the 2 pods

<input type="checkbox"/>	▶	Active	data-collection 	registry-ext.cls.fr:443/odyssea/data-collection/c7.apiv473 + 1 image 1 Pod / Created a month ago / Pod Restarts: 0	1	
<input type="checkbox"/>	▶	Active	data-collection-hangfire 	registry-ext.cls.fr:443/odyssea/data-collection/c7.db12475 + 3 images 1 Pod / Created a month ago / Pod Restarts: 0	1	

- Redeploy the **data-collection** Pods and then the **hangfire** data-collection Pod

<input type="checkbox"/>	▶	Active	data-collection 	registry-ext.cls.fr:443/odyssea/data-collection/c7.apiv473 + 1 image 1 Pod / Created a month ago / Pod Restarts: 0	1	
<input type="checkbox"/>	▶	Active	data-collection-hangfire 	registry-ext.cls.fr:443/odyssea/data-collection/c7.db12475 + 3 images 1 Pod / Created a month ago / Pod Restarts: 0	1	
<input type="checkbox"/>	▶	Active	database-dump-cronjob 	registry-ext.cls.fr:443/odyssea/odyssea-deployment/odysseadump151 3 Pods / Created a month ago / Pod Restarts: 0		
<input type="checkbox"/>	▶	Active	frontend-back 	registry-ext.cls.fr:443/odyssea/frontend/monopv0.8.4 + 1 image 1 Pod / Created 3 months ago / Pod Restarts: 0		
<input type="checkbox"/>	▶	Active	frontend-odyssea 	registry-ext.cls.fr:443/odyssea/frontend/webv0.8.4		

- Wait for pods to restart

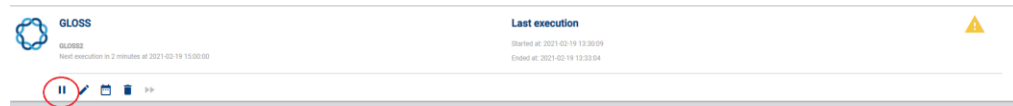
- Check on the interface that the job resumes

Launch the links below in the specified order, there is a bug on this component and we do not recover the jobs otherwise (bypassed the security warning if it appears prob certificate not up to date)

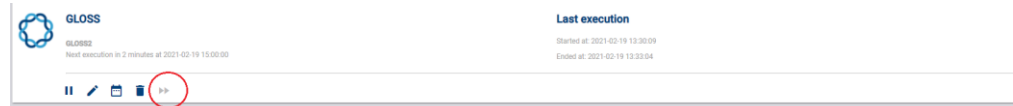
- 1→ <https://swagger.mgmt-svc.cls.fr/swagger/index.html>
- 2→ <https://hangfire.mgmt-svc.cls.fr/swagger/index.html>
- 3→ <https://data-collection.mgmt-svc.cls.fr/jobs>

- From the interface <https://data-collection.mgmt-svc.cls.fr/jobs>

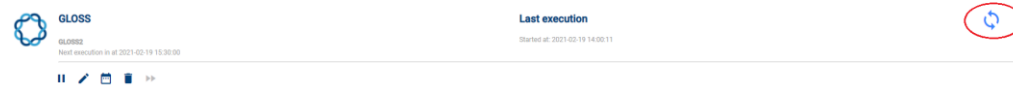
- Restart the **GLOSS** job
- Pause the job if it is already scheduled



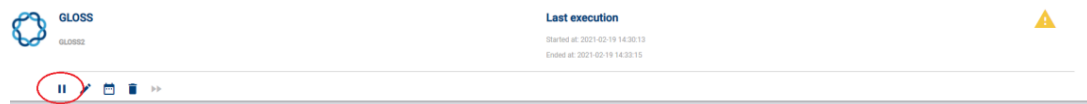
- Pause the job if it is already scheduled



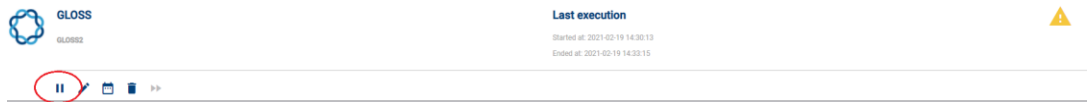
- Normally the update in progress icon should appear



- Refresh the HTML page after 5 minutes, normally the icon attention(warning) must be present (this icon is always visible because for this data we do not recover all)



- Reschedule the job by pressing the pause tab



- Wait 1h-1H30 approximately normally one will have recovered the data thus the alarm will have disappeared. If still nothing, contact the IE so that it investigates

## 4.8 Restoration of the Postgis/ Postgres/ Geonetwork DB

### 4.8.1 Blocked ingestion

#### Important information

The Postgis DB corresponds to the SOS database where the insitu data are saved

- **Purpose of the procedure**

This procedure will make it possible to make a clean of the data in order to restore them from the dump of the schema then the data.

- **Impact and backup solution**

One of the backend DBs is corrupted or data has been lost.

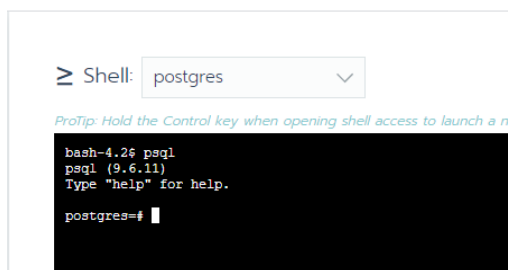
- **Procedure**

#### **Prerequisites**

The DB is no longer accessible or is corrupted.

#### **Steps**

- Clean up the DB
  - Connect to Rancher
  - Connect to the catalog-engine pod and the postgres container
  - Launch psql



```

≥ Shell: postgres
ProTip: Hold the Control key when opening shell access to launch a n
bash-4.2$ psql
psql (9.6.11)
Type "help" for help.

postgres=#
  
```

- Connect to the DB you want to delete (postgis or postgres or geonetwork)

```
postgres=# \c <database name>;
```

```
DO $$
DECLARES
record;
```

```

BEGIN

FOR re IN SELECT *

    FROM pg_namespace ns

    JOIN pg_class c ON ns.oid = c.relnamespace

    JOIN pg_roles r ON r.oid = c.relowner

WHERE

    ns.nspname = 'public' AND

    r.rolname = 'postgis' AND

    ns.nspname NOT IN ('pg_catalog', 'information_schema') AND

    c.relkind = 'r'

LOOP

    RAISE INFO 'Dropping table %', re.relname;

    EXECUTE format('DROP TABLE IF EXISTS %I CASCADE', re.relname);

END LOOP;

END$$;

```

- Execute the following lines to delete the tables from the public schema

```

DO $$

DECLARE

    re record;

BEGIN

FOR re IN SELECT *

    FROM pg_namespace ns

    JOIN pg_class c ON ns.oid = c.relnamespace

    JOIN pg_roles r ON r.oid = c.relowner

WHERE

    ns.nspname = 'public' AND

    r.rolname = 'postgis' AND

    ns.nspname NOT IN ('pg_catalog', 'information_schema') AND

    c.relkind = 'r'

LOOP

    RAISE INFO 'Dropping table %', re.relname;

    EXECUTE format('DROP TABLE IF EXISTS %I CASCADE', re.relname);

END$$;

```

```
END LOOP;

END$$;
```

- Copy the dump to the catalog-engine pod (docker postgres). Remember to activate the right context.

In order to carry out this part of the procedure, you must first have set up the Odyssea context

- Get the pod catalog engine ID

```
root@sel-mokhtari-ramus-instance1:~# kubectl get pods -n <namespace>

NAME READY STATUS RESTARTS AGE
catalog-engine-5dffc87b86-t2l2s 2/2 Running 0 15m
catalog-engine-cronjob-1603971900-6778s 0/1 Completed 0 72m
catalog-engine-cronjob-1603972800-48jpc 0/1 Completed 0 57m
catalog-engine-cronjob-1603975500-bcdvt 0/1 Completed 0 12m
data-collection-6984666796-lcv8f 1/1 Running 0 22h
scheduler-68b48b49b-jqc54 1/1 Running 0 22h
trix-67f9cff9b-pw65s 1/1 Running 0 22h
```

- Copy the DUMPs (schema + data) to the postgres catalog-engine pod

In order to carry out this part of the procedure, you must first have set up the Odyssea context

Example for geonetwork:

```
root@xxxxxxx:~# kubectl cp geonetworkDB_schema_20201028.sql
odysseaqo/catalog-engine-5dffc87b86-
t2l2s:/tmp/geonetworkDB_schema_20201028.sql

root@xxxxxxx:~# kubectl cp geonetworkDB_data_20201028.sql
odysseaqo/catalog-engine-5dffc87b86-
t2l2s:/tmp/geonetworkDB_data_20201028.sql
```

- Restore schema and data



- Log in
- Connect to the catalog-engine pod and the postgres container
- Go to the directory where you saved the DUMP (/tmp for example)
- Run the following command to restore the schema (depending on the schema you want to restore)

```
bash-4.2$ psql -U postgres -d geonetwork -f
geonetworkDB_schema_20201028.sql

bash-4.2$ psql -U postgres -d postgres -f postgresDB_schema_20201028.sql

bash-4.2$ psql -U postgres -d postgis -f postgisDB_schema_20201028.sql
```

- Run the following command to restore the data (depending on the data you want to restore)

```
bash-4.2$ psql -a -U postgres -d geonetwork -f
geonetworkDB_data_20201028.sql

bash-4.2$ psql -a -U postgres -d postgres -f postgresDB_data_20201028.sql

bash-4.2$ psql -a -U postgres -d postgis -f postgisDB_data_20201028.sql
```

The restoration of postgis data can take a little time between 30 min -1 H because there is a large number of observations.

- Restart the pod catalog engine then geonetwork and finally the frontend from the Rancher interface

## 5 Appendix

### 5.1 Organization

#### 5.1.1 Operations Engineer

##### Principal:

Organization	Last name First name	Authorized call ranges (H24, HO..)	Comments
CLS	El Mokhtari Salima	HO/JO	

##### Backup:

Organization	Last name First name	Authorized call ranges (H24, HO..)	Comments
CLS	Calvez Marie-Hélène	HO/JO	

#### 5.1.2 Product Engineer

Organization	Last name First name	Authorized call ranges (H24, HO..)	Comments
CLS	Granier Nicolas	HO/JO	

#### 5.1.3 Developers

Organization	Last name First name	Component
BlueLobster	Keeble Simon	Front End / User Management
Edisoft	Figueiredo Carlos /Pedro Ines	Back SOS
Hidromod	Galvão Pedro	In Situ Data Collection
CLS	Marty Sylvain / Doumerc Jérôme	Product Factory /MOTU/OGC WS

## 5.2 K8S / Rancher controls

### 5.2.1 Connection to the K8S PROD cluster

- Please connect to the computer where the kubectl client is installed
- Display the list of available contexts

```
root@xxxxxxxx:~$ kubectl config get-contexts

CURRENT NAME CLUSTER AUTHINFO NAMESPACE
* k8s-fs-qt1 u-eqtu3gi2lg
    k8s-prod1
    k8s-prod1-fqdn k8s-prod1
```

- Define the context you want to use, which is the production context

```
root@xxxxxxxx:~$ kubectl config use-context k8s-prod1

Switched to context "k8s-prod1".
```

- You can now run commands via [kubectl](#) in the Kubernetes cluster of prod that will allow you to inspect and manage the resources of the cluster and consult the log of the Odyssea application. An example of command is given below `kubectl get pods -n < namespace>` that allows to display the list of pods of the Odyssea application

```
root@xxxxxxxx:~$ kubectl get pods -n odyssea

NAME READY STATUS RESTARTS AGE
catalog-engine-77b9b5bd4-tp94f 2/2 Running 0 31d
catalog-engine-clean-h2-base-cronjob-1580086620-wkftf 0/1 Completed 0 7h34m
catalog-engine-clean-h2-base-cronjob-1580086680-8vz8r 0/1 Completed 0 7h33m
catalog-engine-clean-h2-base-cronjob-1580086740-4nzkh 0/1 Completed 0 7h32m
catalog-engine-clean-processed-dir-cronjob-1580086620-9tmc7 0/1 Completed 0 7h34m
catalog-engine-clean-processed-dir-cronjob-1580086680-92xv6 0/1 Completed 0 7h33m
catalog-engine-clean-processed-dir-cronjob-1580086740-fcpdn 0/1 Completed 0 7h32m
catalog-engine-cronjob-1580108400-5htjx 0/1 Completed 0 91m
catalog-engine-cronjob-1580111100-zn2en 0/1 Completed 0 46m
catalog-engine-cronjob-1580112000-5zd6v 0/1 Completed 0 31m
```

## 5.2.2 Update the Config file to access the K8SPROD1 cluster

### 5.2.2.1 Update of the config file to access the k8sprod1 cluster (Odyssea context)

#### **Important information**

It is assumed that the kubectl executable is already configured, otherwise contact the IE

- Get the new configuration file from gitlab ope
- Open a power shell window in order to identify where the kubeconfig file is copied on the machine that will allow you to access the cluster

```
PS C:\Users> cd $HOME  
  
PS C:\Users\XXXXX> cd . kube  
  
PS C:\Users\XXXXX\.kube> pwd  
  
Path  
  
----  
  
C:\Users\xxxx\.kube
```

- Open a Windows window to the . kube folder
- Delete the old existing config file and paste the file retrieved from git or the one linked in this page if it is not too old (renaming it config if needed)
- Open a new PowerShell window and test the following commands

```
PS C:\Users\xxxxx\.kube> kubectl config get-contexts  
  
PS C:\Users\xxxxx\.kube> kubectl config use-context k8s-prod1  
  
Switched to context "k8s-prod1".  
  
PS C:\Users\xxxxx\.kube> kubectl get pods -n odyssea  
  
NAME READY STATUS RESTARTS AGE  
catalog-engine-5f9694754b-6sxcz 2/2 Running 0 172m  
catalog-engine-cronjob-1610539200-4wlrf 0/1 Completed 0 102m  
catalog-engine-cronjob-1610541900-xjt99 0/1 Completed 0 57m  
catalog-engine-cronjob-1610542800-8mc8n 0/1 Completed 0 42m  
  
TEST CONNECTION TO A POD:  
  
PS C:\Users\xxxxx\.kube> kubectl exec -it nrpe-cf4d866f9-kjdzv -n odyssea bash
```

```
root@nrpe-cf4d866f9-kjdz:/# exit  
exit
```

### 5.2.2.2 Incorrect Odyssea context

It happened that the Odyssea context was no longer the right one, this type of error appeared

```
root@xxxxxxxx:~$ kubectl get pod -n odyssea  
error: You must be logged in to the server (the server has asked for the client to provide credentials)
```

Contact the IE to contact them so that they can delete and recreate this user and provide you with a new kubeconfig

### 5.2.3 Stop Restart & Useful Commands

#### **Important information**

In order to be able to carry out this procedure, you must first have set up the Odyssea context

- **Relaunching a POD**

Deleting a pod will automatically recreate it

- Get the id of the Pod you want to redeploy

```
root@sel-mokhtari-ramus-instance1:~$ kubectl get pods -n odyssea  
  
NAME READY STATUS RESTARTS AGE  
catalog-engine-5978987d49-wrtzn 2/2 Running 0 9d  
catalog-engine-clean-h2-base-cronjob-1586912220-mwq4j 0/1 Completed 0 13h  
data-collection-b89855548-f4hn8 1/1 Running 0 7d8h  
data-collection-hangfire-5cdf98775c-57cm7 2/2 Running 0 7d8h  
frontend-back-65fdd4bd9d-429nd 2/2 Running 0 7d8h  
frontend-odyssea-f8c5cd8b7-bq7vt 1/1 Running 64 7d8h  
geonetwork-68759677b5-dvv56 1/1 Running 0 12d  
nrpe-7968846d77-5fpf8 1/1 Running 0 6d1h
```

- Delete this Pod

```
root@xxxxxxxx:~# kubectl delete pod <pod name for example nrpe-7968846d77-5fpf8 > -n <
namespace>

pod "nrpe-7968846d77-5fpf8" deleted
```

#### Important information

Wait until the shell gives you back your hand

- List the available Pods and check that the Pod you have deleted is in a Running state as can be seen in the example below the nrpe Pod is in a 'Running' state and you can see that it has been recreated as its 'age' is now 2m42s whereas it was previously 6 days

```
root@sel-mokhtari-ramus-instance1:~s# kubectl get pods -n odyssea

NAME READY STATUS RESTARTS AGE
nrpe-7968846d77-9pk2k 1/1 Running 0 2m42s
```

- Recovery of available contexts

```
kubectl config get-contexts
```

- Use of a particular context

```
kubectl config get-contexts
```

- Creating a cronjob

```
kubectl create job --from=cronjob/<name cronjob> <name of choice> -n <namespace>
```

- Creation of secret key registry GITLAB

```
kubectl create secret docker-registry <name> --docker-server=registry-ext.cls.en:443 --docker-
username=< username> --docker-password=<password> --docker-email=<email>
```

- Creating an element on the cluster

```
kubectl create -f < file name (config map, ingress,service, deployment) -n < namespace>
```

- Updating an element on the cluster

```
kubectl apply -f < file name (config map, ingress,service, deployment) -n < namespace>
```

- **Bash**

```
kubectl exec -it < pod> -c <container if more than one > -n < namespace> --bash
```

- **Copy from local station to a pod container**

```
kubectl cp <local file> < namespace>:< pod>:<path in container> -c <container> -n < namespace>
```

- **Listing of deployed services**

```
kubectl get svc -n <namespace>
```

- **Listing of deployed pods**

```
kubectl get pods -n <namespace>
```

- **Removal of cronjob errors**

```
kubectl delete job $(kubectl get jobs -n <namespace> | awk '$3 ~ 0' | awk '{print $1}') -n <namespace>
```

```
kubectl delete job $(kubectl get jobs -n <namespace> | awk '$3 ~ 1' | awk '{print $1}') -n <namespace>
```

```
kubectl delete job $(kubectl get jobs -n <namespace> | awk '{print $1}') -n <namespace>
```

- **Deleting pods with a terminating status**

```
for p in $(kubectl get pods -n <namespace> | grep Terminating | awk '{print $1}');  
do kubectl delete pod -n <namespace> $p --grace-period=0 --force;done
```

#### 5.2.4 Viewing log files

To be able to access the pod that contains the logs you will need to have set up the Odyssea context first

Odyssea backend components logs have been integrated to the NRPE pod. To view them, you just have to connect to the NRPE pod and refer to the path described in the table of paths to these files (below)

- Check that you have activated the right context

```
root@xxxxxxx: kubectl config current-context
k8s-prod1
```

- Retrieve nrpe pod id and connect to nrpe pod

```
root@xxxxxxx: kubectl get pods -n odyssea
root@xxxxxxx: kubectl exec -it nrpe-XXXXXXX bash
```

- Refer to the table below to identify the folder you are looking for

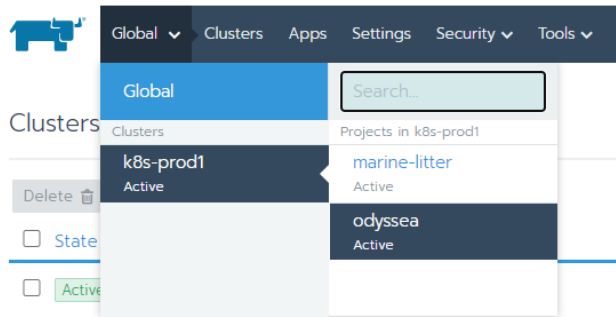
Component	Description of the log	Path
DataCollection	Log of file download from FTP CMEMS and via HTTP for GLOSS	/Download
	Directory containing the list of recovered NetCDF by Data Collection before ingestion pushes them into BDD	/Export
Catalog-engine	Engine log ingestion	/opt/Edisoft/log/
	Folder containing NetCDF files that have been successfully pushed to the database	/opt/Processed
	Folder containing NetCDF files in error	/opt/Processed_with_ERROR

## 5.2.5 Introduction to Rancher

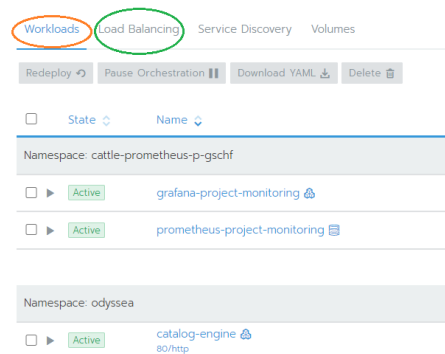
### 5.2.5.1 Connecting to Rancher

- Connect to the Rancher interface <https://rancher-p1.clouds.cls.fr/> using the user clsops
- Select the application (namespace) Odyssea





- You will reach the following interface where two subsections are of general interest
  - Workloads → Application PODs
  - Load Balancings → which list the ingress (HTTP link)

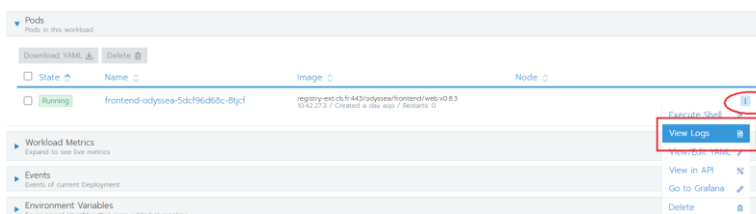


### 5.2.5.2 View PODS logs

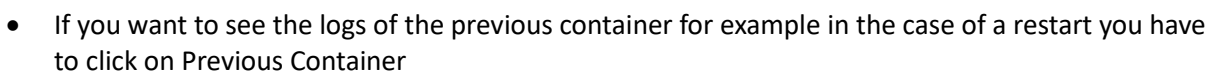
- Click on the pod you wish to investigate

<input type="checkbox"/>	Active	frontend-back	registry-ext.dls.fr:443/odyssea/frontend/monopv0.8.3 • 1 image 1 Pod / Created a day ago / Pod Restarts: 0	1	ⓘ
<input type="checkbox"/>	Active	frontend-odyssea	registry-ext.dls.fr:443/odyssea/frontend/webv0.8.3 1 Pod / Created a day ago / Pod Restarts: 0	1	ⓘ
<input type="checkbox"/>	Active	geonetwork	registry-ext.dls.fr:443/odyssea/catalog3.10.3-1 • 2 images 1 Pod / Created a day ago / Pod Restarts: 0	1	ⓘ

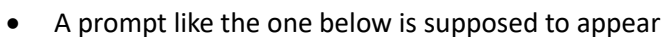
- Right click on Views logs



- You are now viewing the container logs





- Click to the right of the pod for which you want to run a bash window



#### 5.2.5.4 Follow an HTTP link from the application

- Click on the Load Balancing section and then on the link you want to test (it is possible that you need to complete the URL as for geoNetwork see complete URL in Accounts and passwords section)

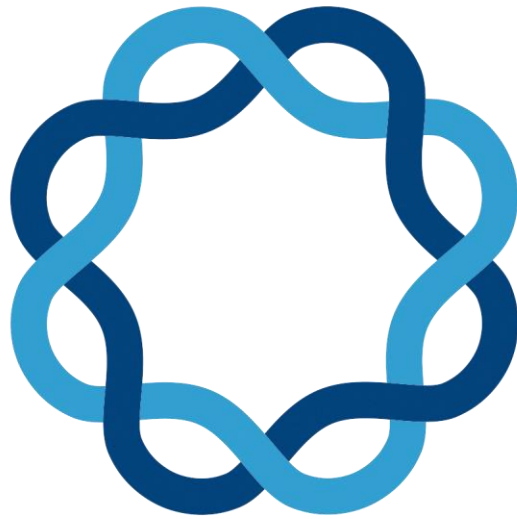
Workloads **Load Balancing** Service Discovery Volumes

Download YAML  Delete 

☐ State ☐ Name ☐ Targets

Namespace: odyssea

<input type="checkbox"/>	Active	data-collection L7 Ingress	data-collection.mgmt-svc.cls.fr > data-collection
<input type="checkbox"/>	Active	geonetwork L7 Ingress	geonetwork.svc.groupcls.com > tomcat-geonetworks
<input type="checkbox"/>	Active	hangfire L7 Ingress	hangfire.mgmt-svc.cls.fr > hangfire
<input type="checkbox"/>	Active	marinomica L7 Ingress	<u>marinomica.com/</u> > frontend-odyssea
<input type="checkbox"/>	Active	nginxfrontal L7 Ingress	wps.odyssea.mgmt-svc.cls.fr / > service-nginxfrontal
<input type="checkbox"/>	Active	sos-server-52north-fro... L7 Ingress	sos-server-52north-frontend.svc.groupcls.com / > tomcat-sos
<input type="checkbox"/>	Active	swagger L7 Ingress	swagger.mgmt-svc.cls.fr > swagger



# ODYSSEA