OpenWIS v5

A long-term vision
for the evolution of the WIS

Met Office

Toulouse, 20-23 March 2017

Draft Recommendation 5.5(1)/1 (CBS-16)

”WIS 2.0 will provide users with seamless access to diverse information from a wide range of sources and will enable weather, water and climate information to be related to socioeconomic and other application contexts. Through an open eco-system of tools, applications and services, WIS 2.0 will allow all information providers to manage, publish and share their data, products and services and will allow all users to develop value added services and new products.”

Draft Recommendation 5.5(1)/1 (CBS-16)
WIS 2.0 strategy

Accessibility: Enhance data collection, high-volume, reduced complexity

Interoperability: Industry standard formats

Visibility: Data visible to government, commerce and citizens

Utility: Exploit meteorological data in context with data from other domains

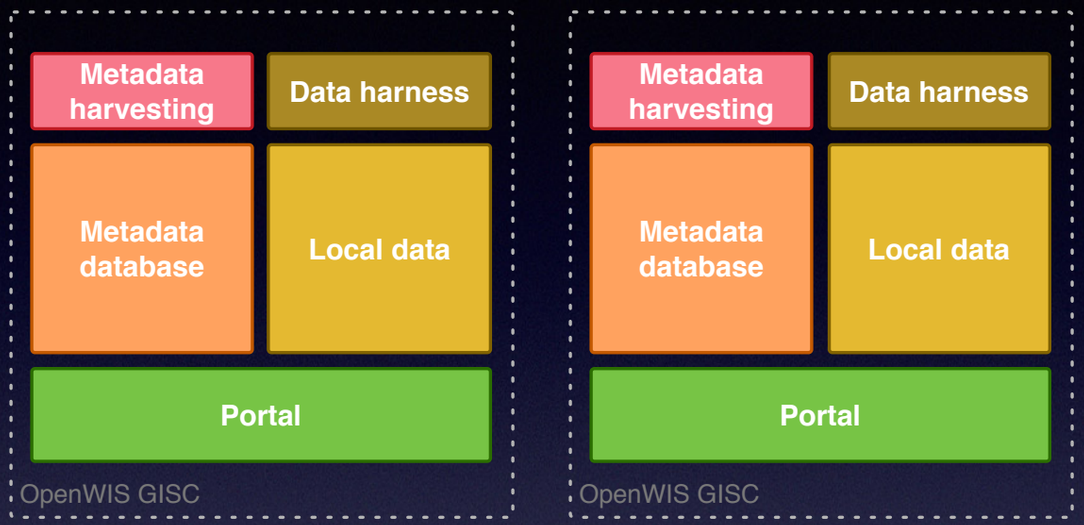
Draft Recommendation 5.5(1)/1 (CBS-16)
WIS 2.0 strategy

Reliability: Safe with guaranteed performance

Cost effectiveness: Shared components, economy of scale

Capacity-building: Training services

OpenWIS v3
Typical GISCs



Autonomous systems

Nothing shared

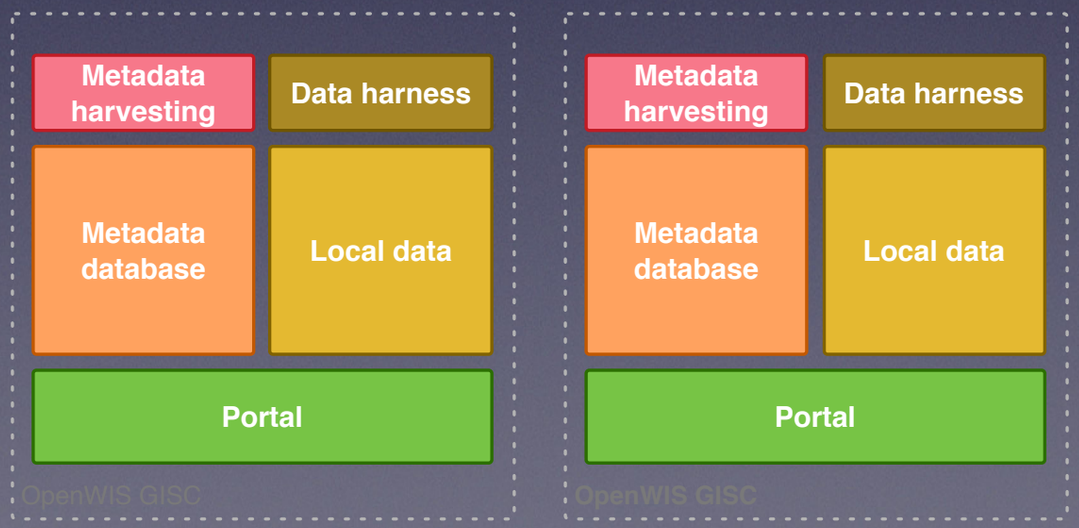
Large data

=> Large infrastructure

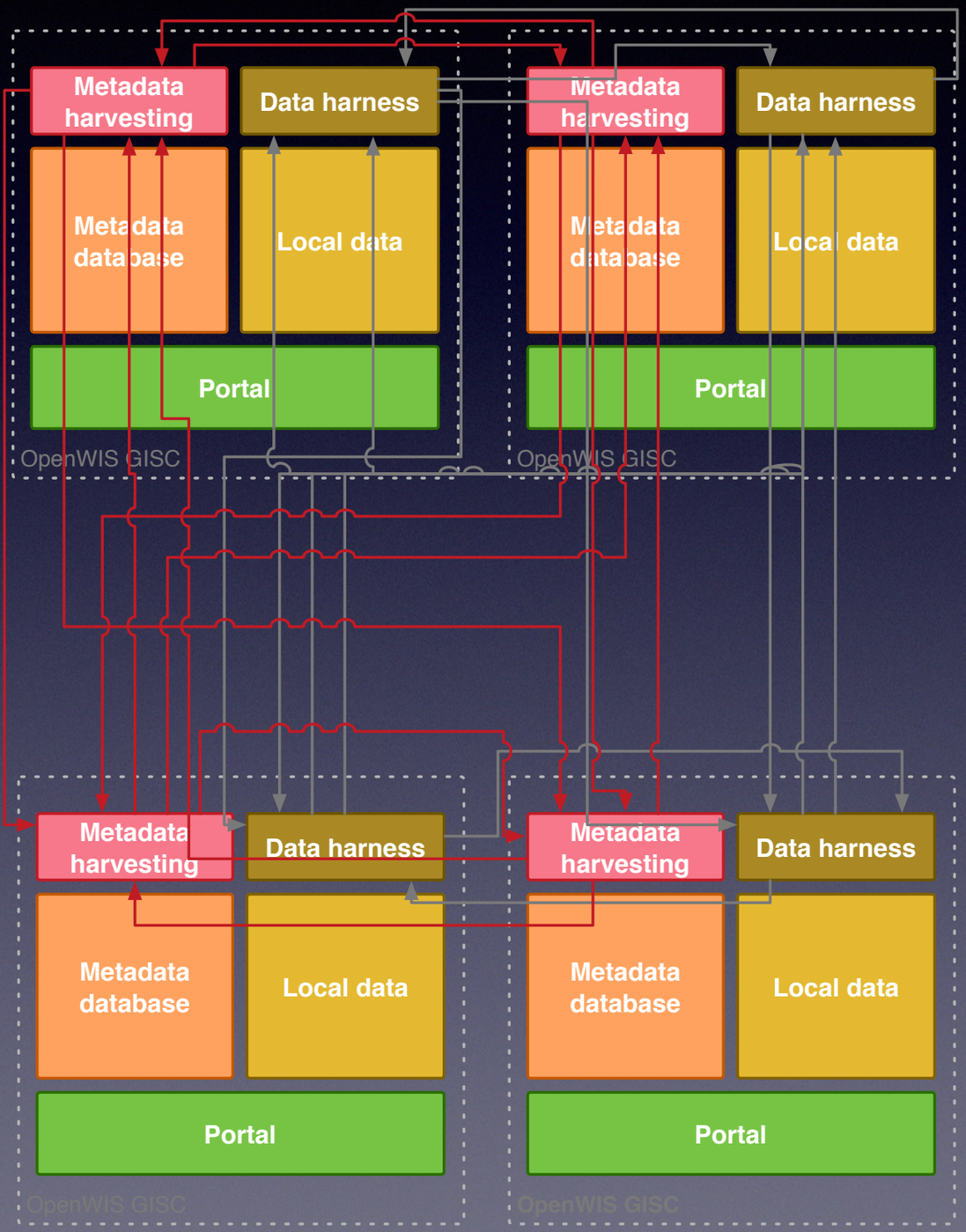
=> Long processing

Complex installation

Clustering complexity



OpenWIS v3
Data exchange



High-traffic

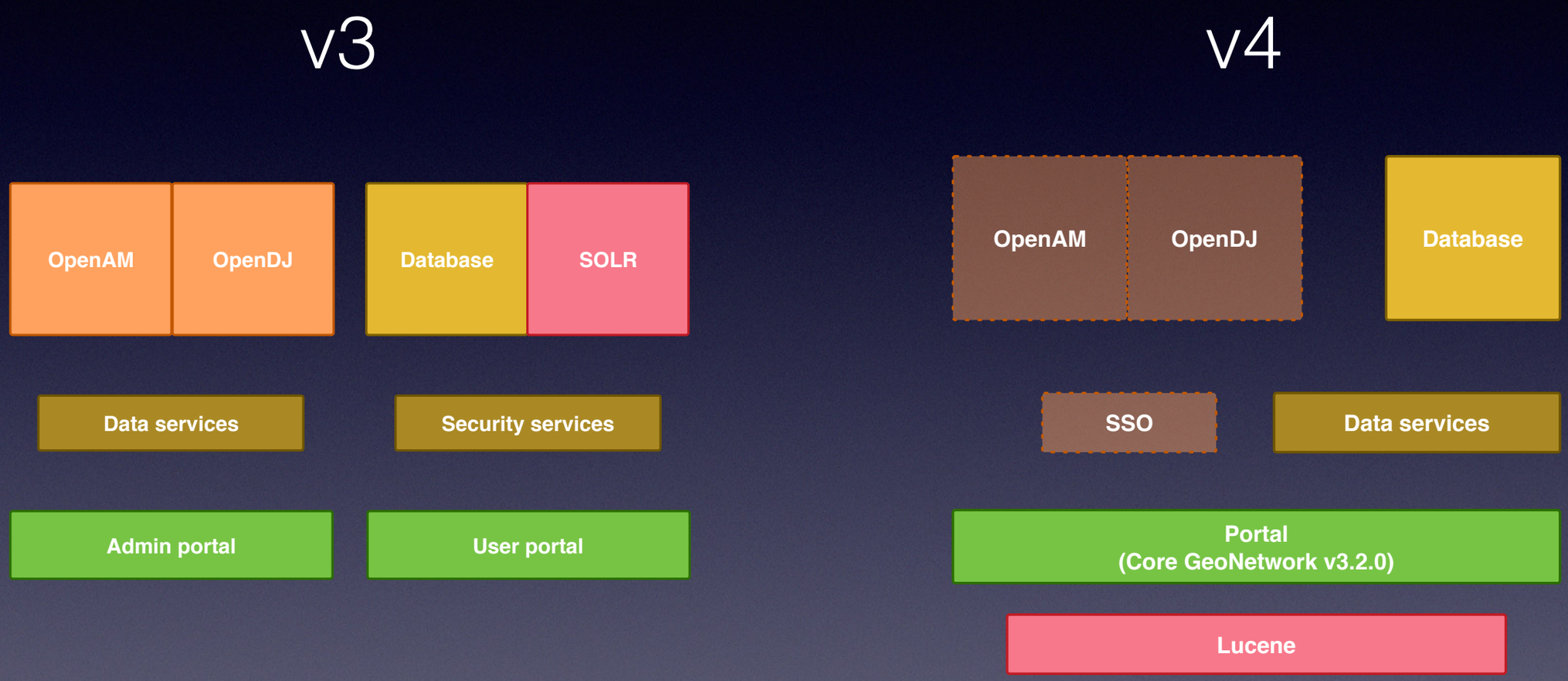
GISCs become data distribution hubs

Same data processed again and again

Slow harvesting

No discovery mechanism

OpenWIS v3 to v4
Overview



Better UI but no changes in terms of data processing, bandwidth usage...

OpenWIS v4

Harvesting performance - out of the box

Embedded H2
Local shapefile

Very slow when data accumulates

Practically unusable after 100K records

PROVIDER	GEONETWORK 3.1.0-SNAPSHOT (develop/991048eb763a39ba3cf4cf4d88d23f19e391ea1b)			
	set (iso19139)	records	time (sec)	rate (r/sec)
France http://wispi.meteo.fr/openwis-user-portal/srv/oaipmh	WIS-GISC-TOULOUSE	2947	674	4,37
Moscow http://meta.gisc-msk.wis.mecom.ru/openwis-portal/srv/ru/oaipmh	WIS-GISC-MOSCOW	3429	1837	1,87
Melbourne http://wis.bom.gov.au/openwis-user-portal/srv/oaipmh	WIS-GISC-MELBOURNE	1113	454	2,45
Brasilia http://gisc.inmet.gov.br/oai/provider	WIS-GISC-BRASILIA	557	836	0,67
Tokyo http://www.wis-jma.go.jp/meta/oaiprovider.jsp	WIS-GISC-TOKYO	**Nassos Michas:** Did not start fetching (~30'). Could not stop it, had to restart Jetty.		
Beijing http://oai.dwd.de/oai/provider	WIS-GISC-BEIJING	61600	69388	0,89
Germany http://oai.dwd.de/oai/provider	WIS-DE	30245	65094	0,46
Seoul http://gisc.kma.go.kr/openwis-user-portal/srv/oaipmh	WIS-GISC-SEOUL	**Nassos Michas:** WIS-GISC-SEOUL was only returning 1 record		
Exeter http://wis.metoffice.gov.uk/openwis-user-portal/srv/oaipmh	WIS-GISC-EXETER	17033	Had to restart it multime times, so no accurate readings	
France http://wispi.meteo.fr/openwis-user-portal/srv/oaipmh	WIS-DCPC-EUMETSAT	21	80	0,26
France http://wispi.meteo.fr/openwis-user-portal/srv/oaipmh	WIS-GISC-JEDDAH	248	724	0,34
	TOTAL:	116945		

OpenWIS v4
Harvesting performance - PostGIS

Embedded H2
PostGIS shapefile

Better but still slow

GEONETWORK 3.1.0-SNAPSHOT (develop/2f56642b5e71ddd946ba8ca4a1d52a58323571d9) - Embedded Jetty/PostGIS (8 threads)/Java8			
set (iso19139)	records	time (sec)	rate (r/sec)
WIS-GISC-TOULOUSE	3242	576	5,63
WIS-GISC-MOSCOW	3429	1393	2,46
WIS-GISC-MELBOURNE	1113	It was very slow, had to restart multiple times.	
WIS-GISC-BRASILIA	557	321	1,74

After some long source code profiling sessions...

=> Sequential HTTP harvesting and indexing does not scale!

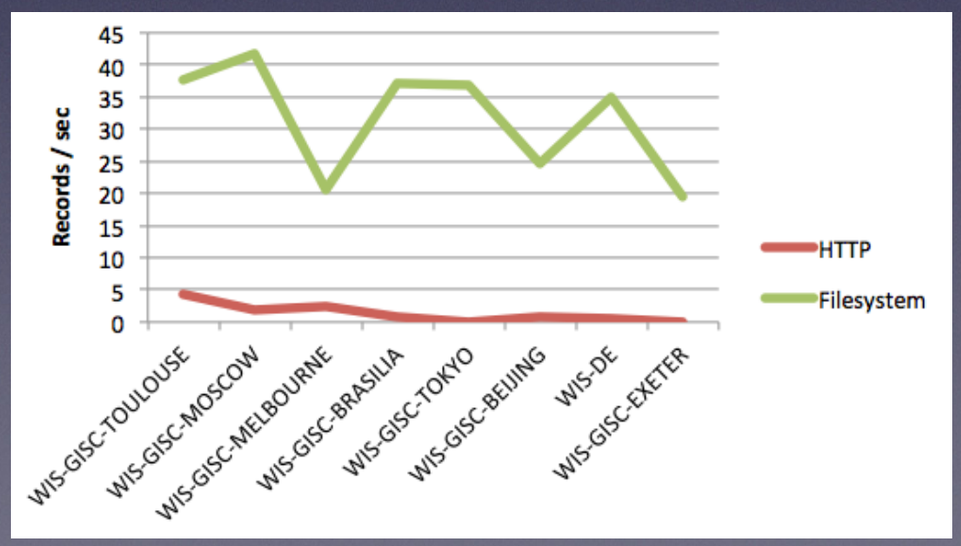
OpenWIS v4

Harvesting performance - offline harvesting

Fetch all metadata offline

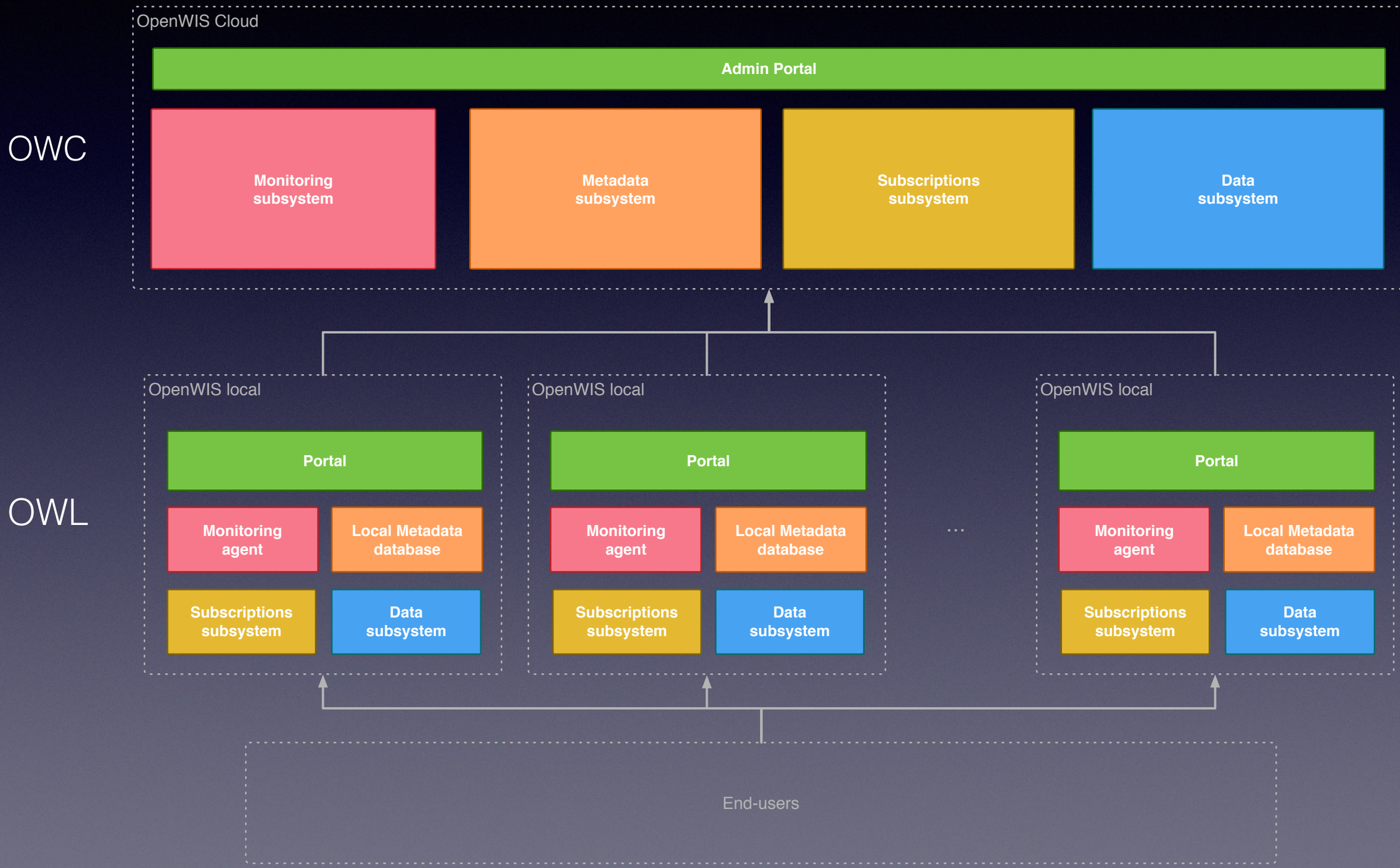
Perform filesystem-based indexing

Seems to be the only option for prod



GEONETWORK 3.1.0-SNAPSHOT (develop/2f56642b5e71ddd946ba8ca4a1d52a58323571d9) - Embedded Jetty/PostGIS (8 threads)/Java8 - Filesystem import			
set (iso19139)	**records**	**time (sec)**	**rate (r/sec)**
WIS-GISC-TOULOUSE	3242	86	37,70
WIS-GISC-MOSCOW	3429	82	41,82
WIS-GISC-MELBOURNE	1113	54	20,61
WIS-GISC-BRASILIA	557	15	37,13
WIS-GISC-TOKYO	24419	664	36,78
WIS-GISC-BEIJING	61600	2504	24,60
WIS-DE	30415	869	35,00
WIS-GISC-SEOUL	WIS-GISC-SEOUL was only returning 5 records		
WIS-GISC-EXETER	17033	880	19,36

OpenWIS v5
Architecture overview



OpenWIS v5
Architecture overview

Why a "hybrid" cloud?

All components modularised as *containers*

Pick'n'mix deployment approach

Use/Deploy what you need

Fully-automated, *single line* deployment

Fully-automated clustering and high-availability

OpenWIS v5
Cloud

Harvesting and Indexing at OWC

Heavy processing at OWC

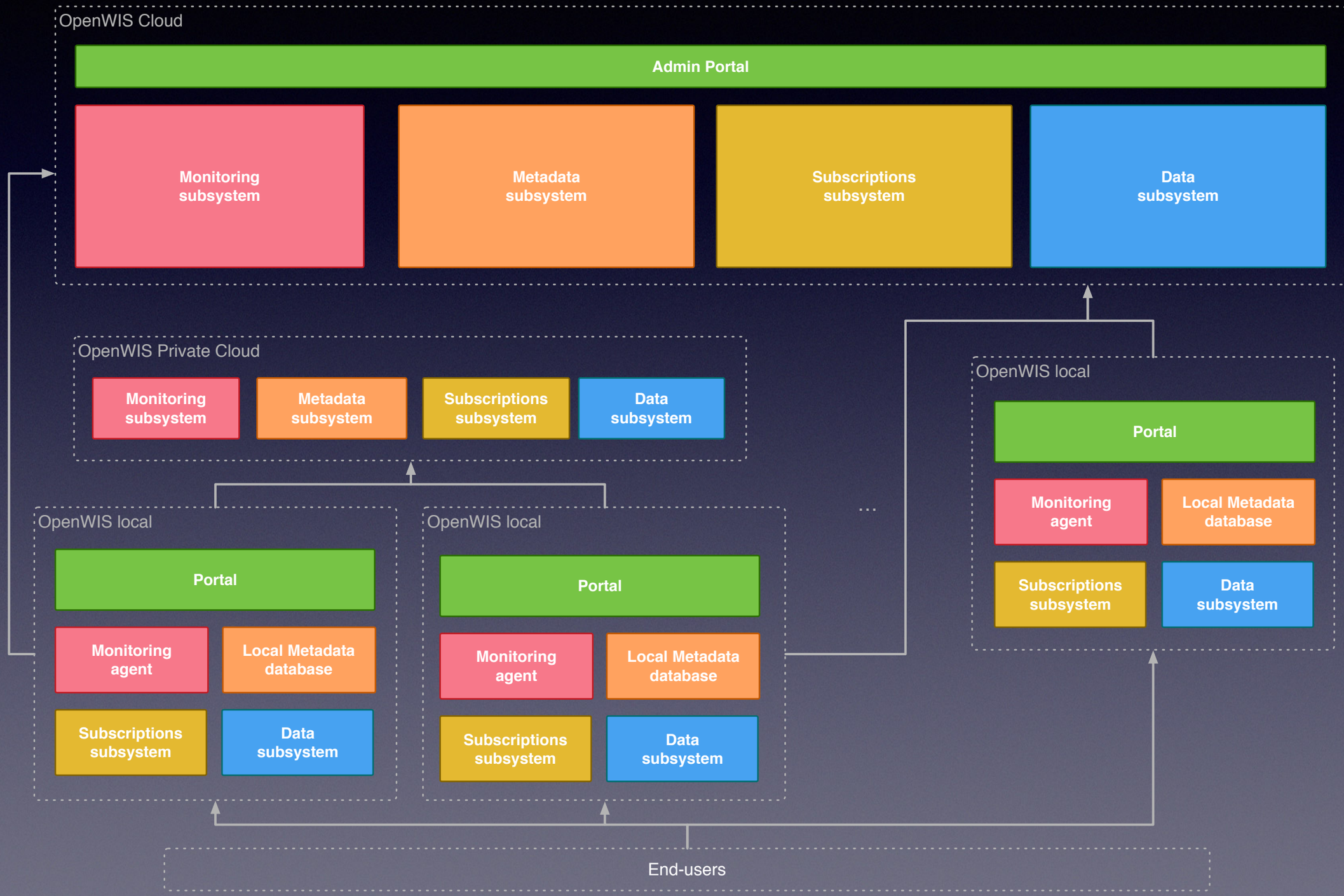
Data storage (e.g. GTS) at OWC

OWL to maintain localised/private data

Open Access to data if published at OWC

How to support private data exchange?

OpenWIS v5
Private cloud



OpenWIS v5
Private cloud

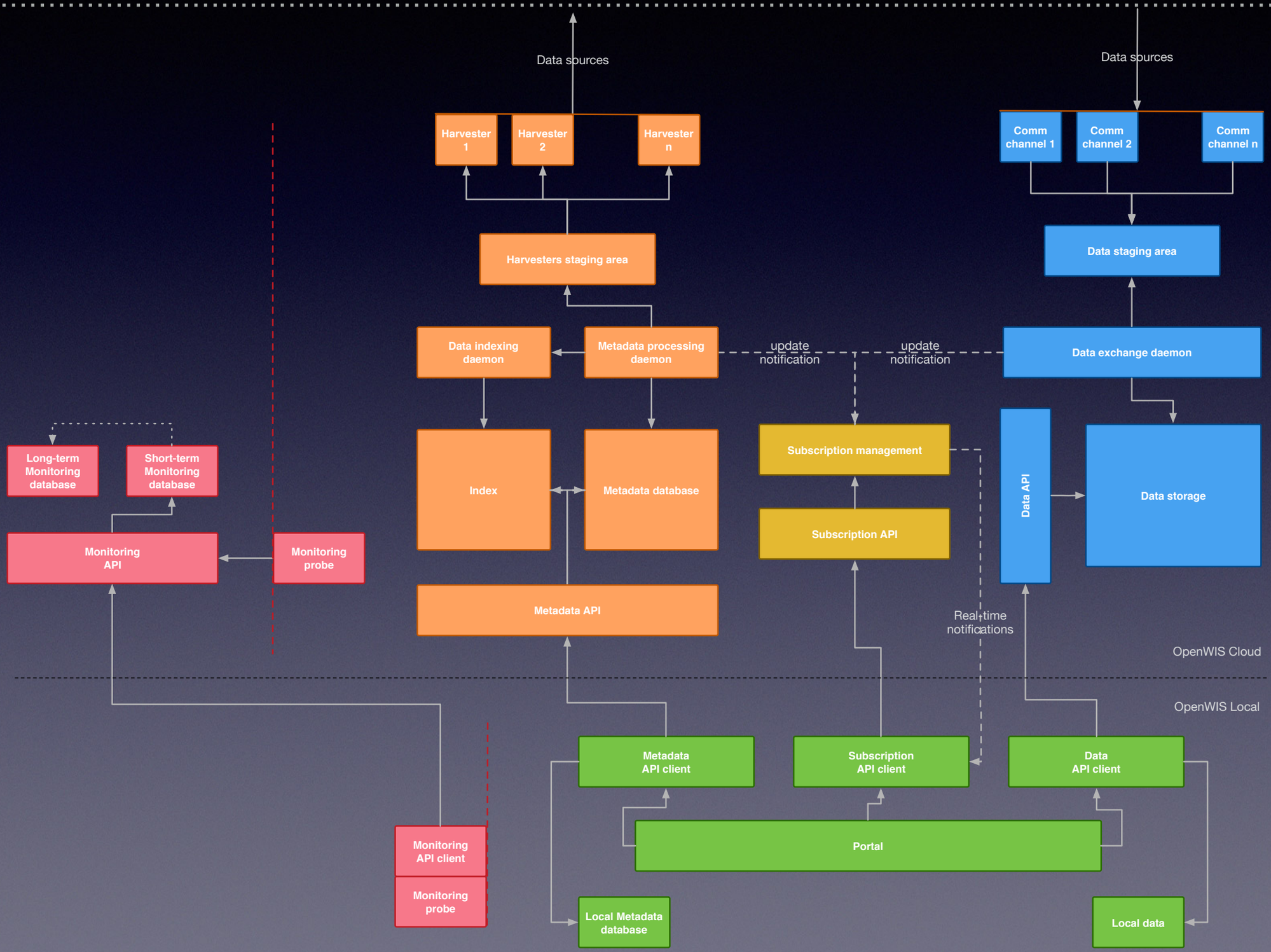
OWPC allows private data exchange

Easy to setup, minimal resource usage

OWL can consume services from OWC and/or zero or more OWPCs

OWPC provides efficiency even at small scale

OpenWIS v5
High-level architecture



OpenWIS v5
High-level architecture

Fully asynchronous harvesting and indexing

Message-driven data/metadata architecture

Subscriptions mechanism based on messaging

State of the art technical stack:
Angular2, ActiveMQ, Apache Camel, Apache CXF, OSGi via Apache Karaf, Elasticsearch, JEE, Docker, Postgres

OpenWIS v5
Cloud infrastructure agnostic

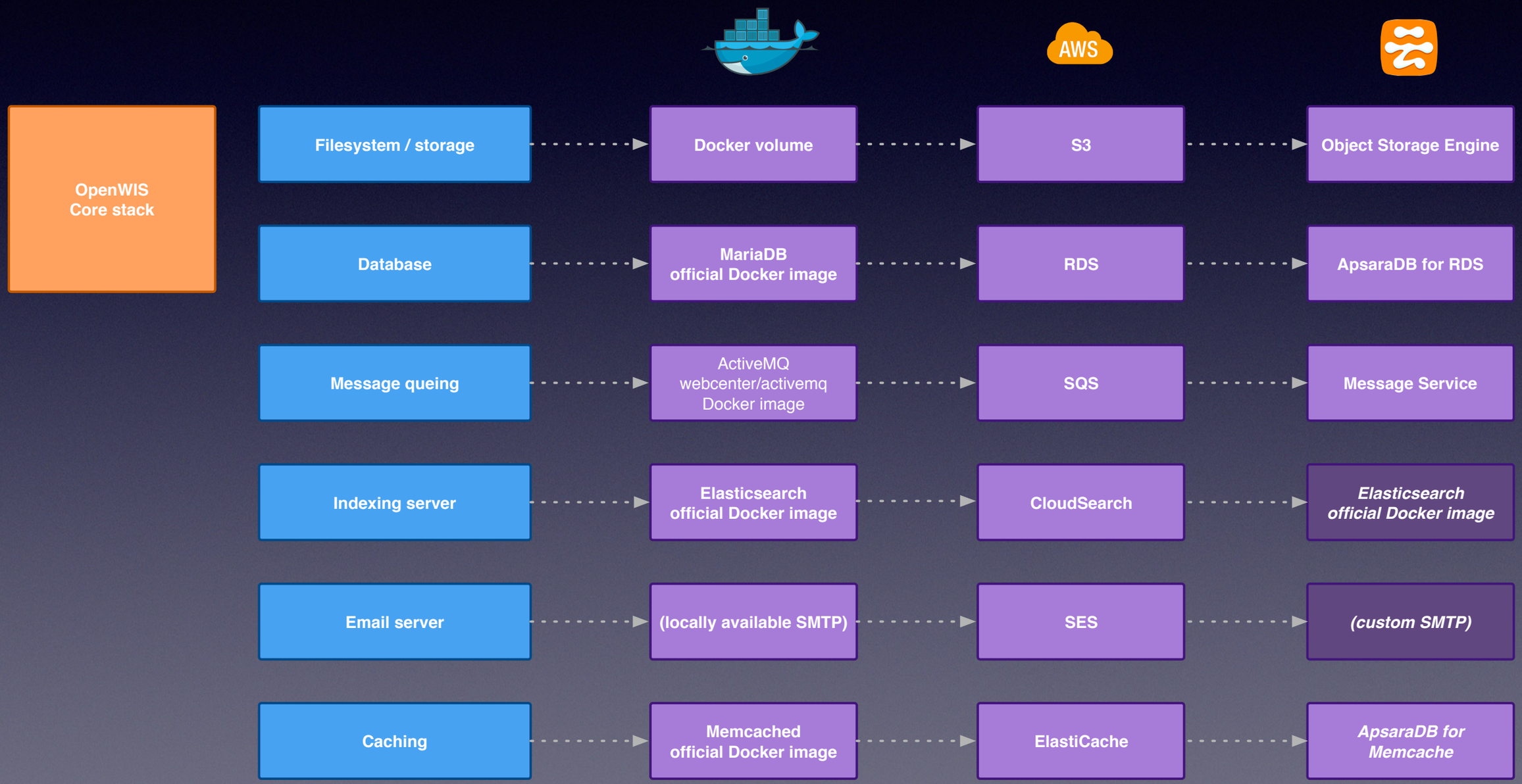
Because of containers-based architecture members are free to decide what to use

An "all inclusive" configuration will be provided for quick and easy setup

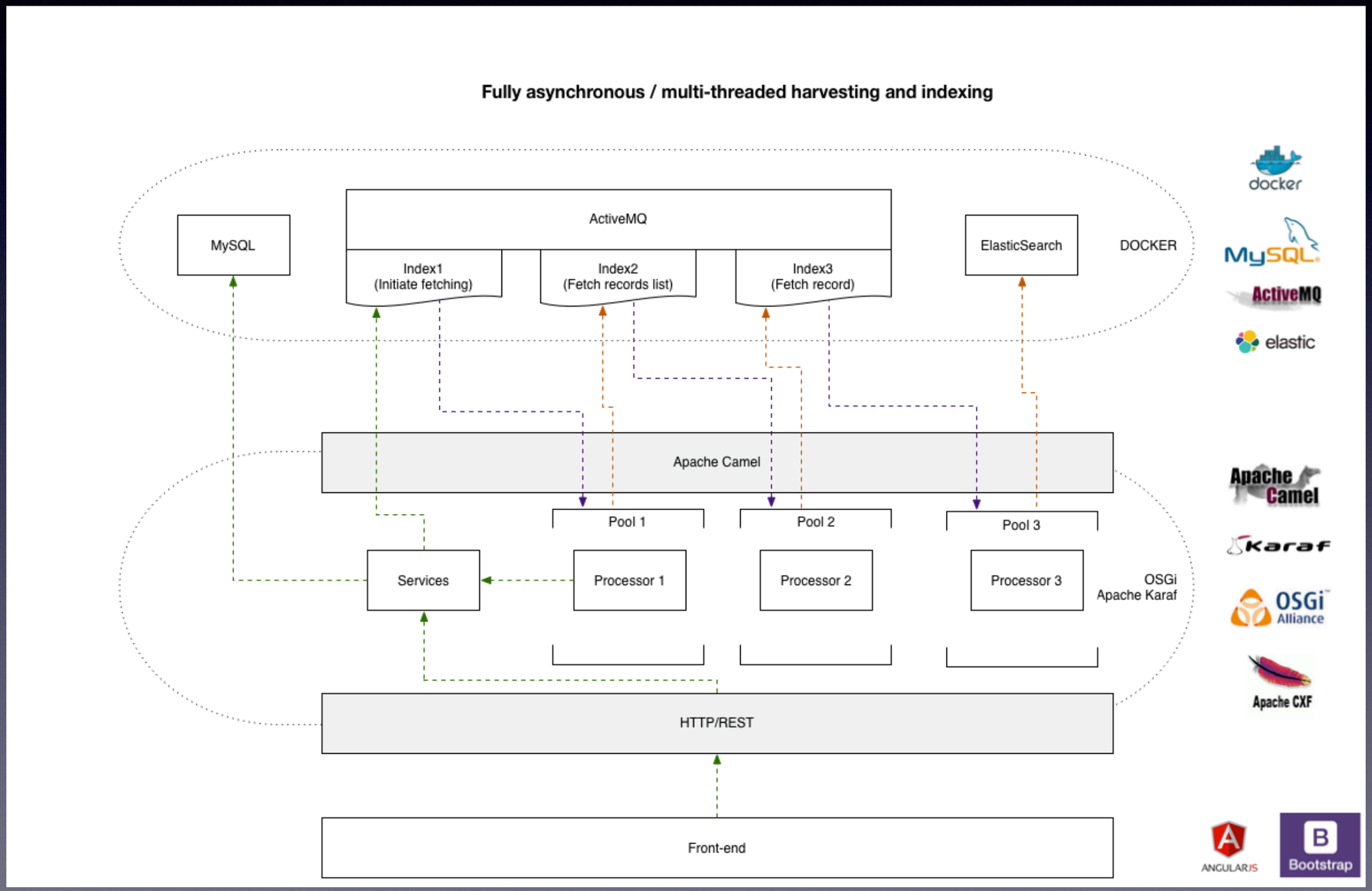
Individual components can be replaced with cloud-specific counterparts

Examples...

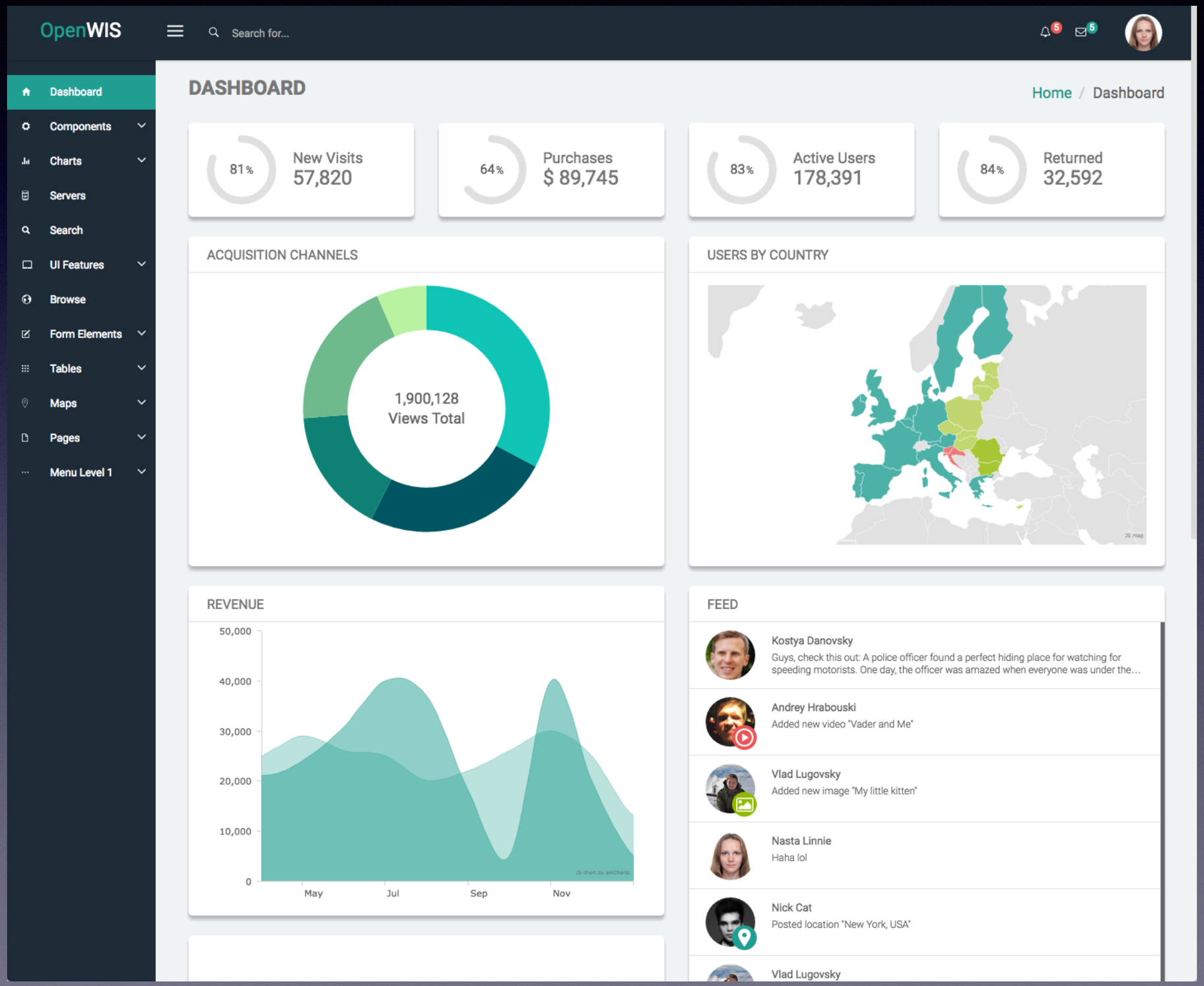
OpenWIS v5
Cloud infrastructure agnostic



OpenWIS v5
PoC - Architecture

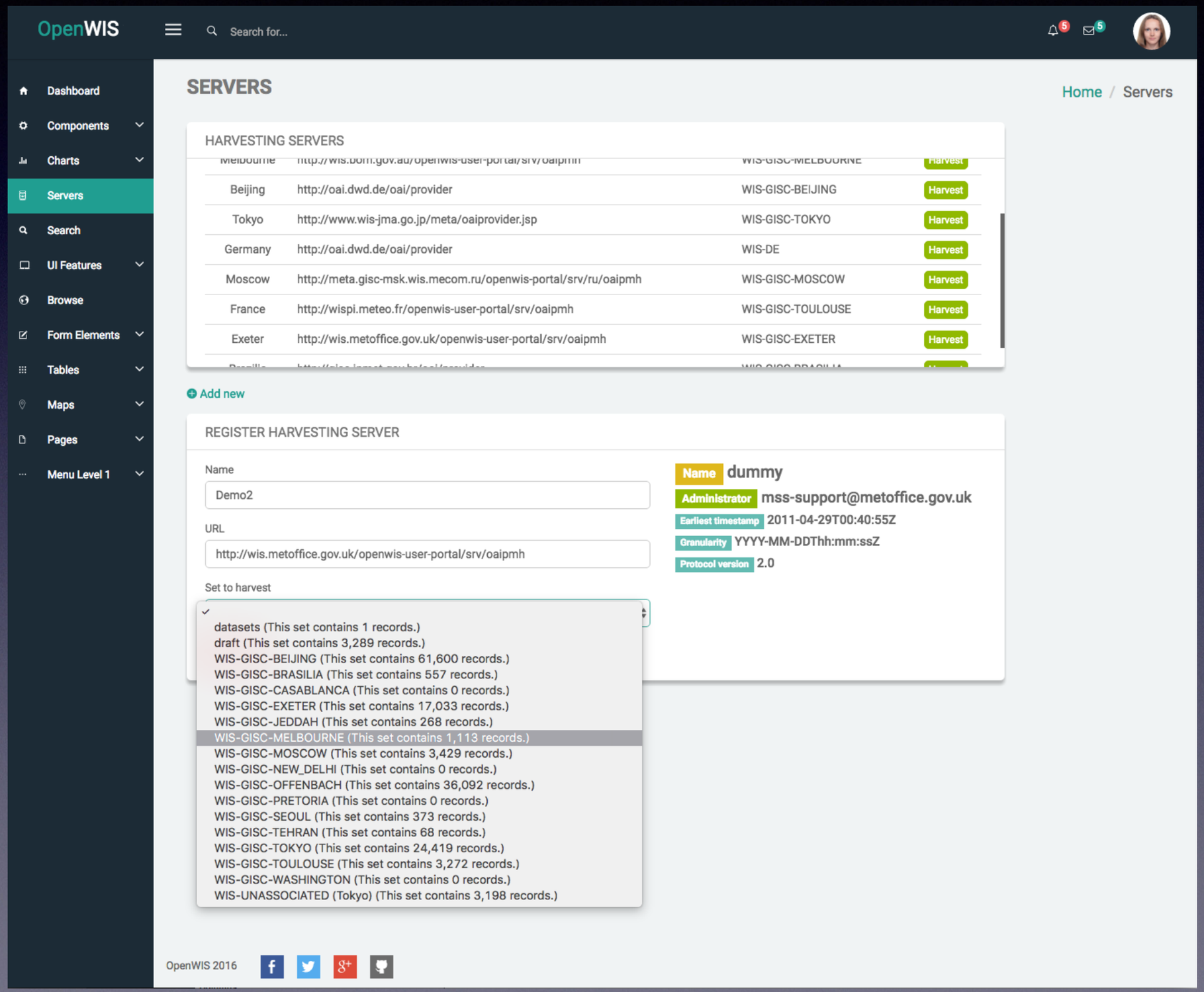


OpenWIS v5
PoC - Home

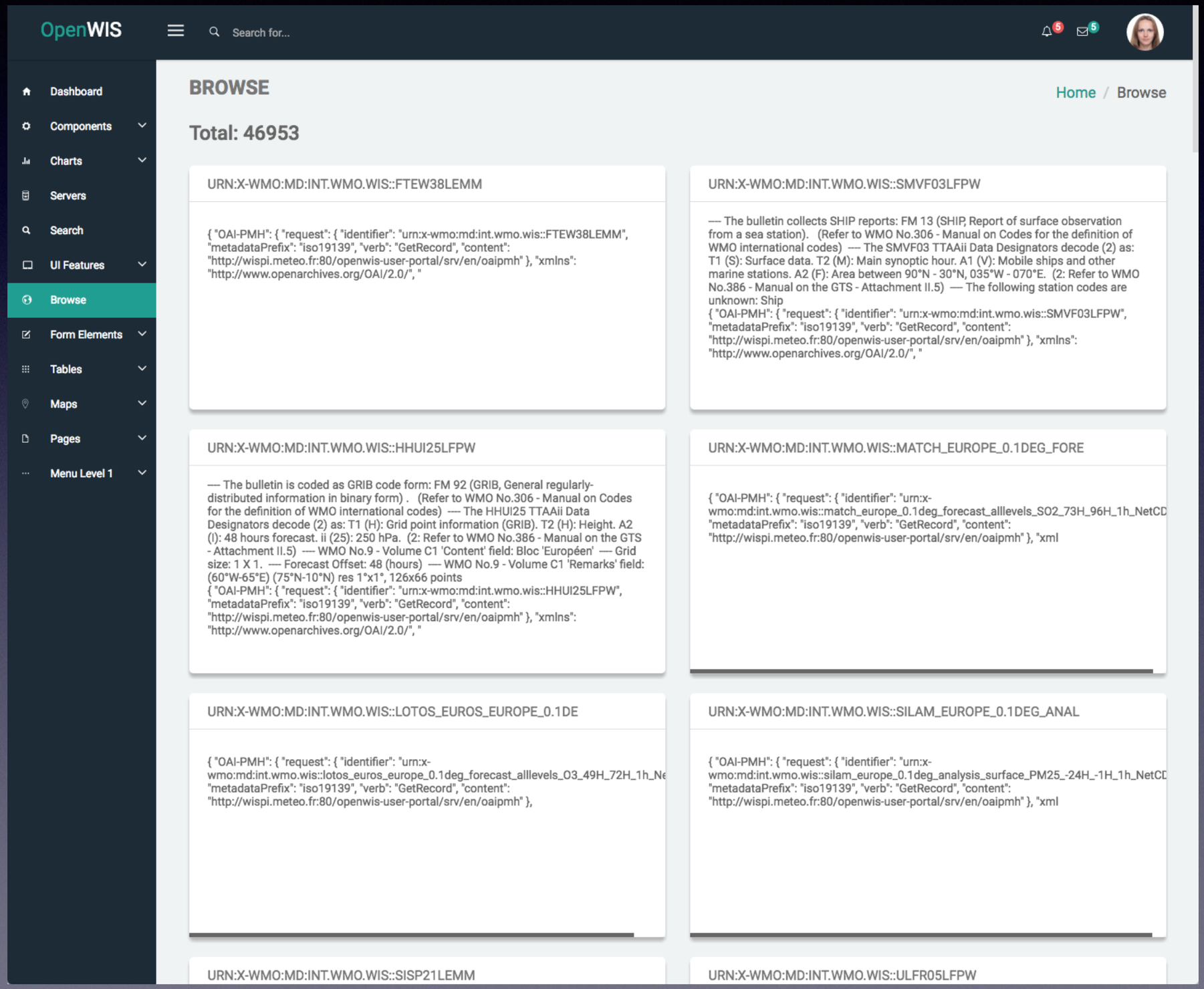


OpenWIS v5

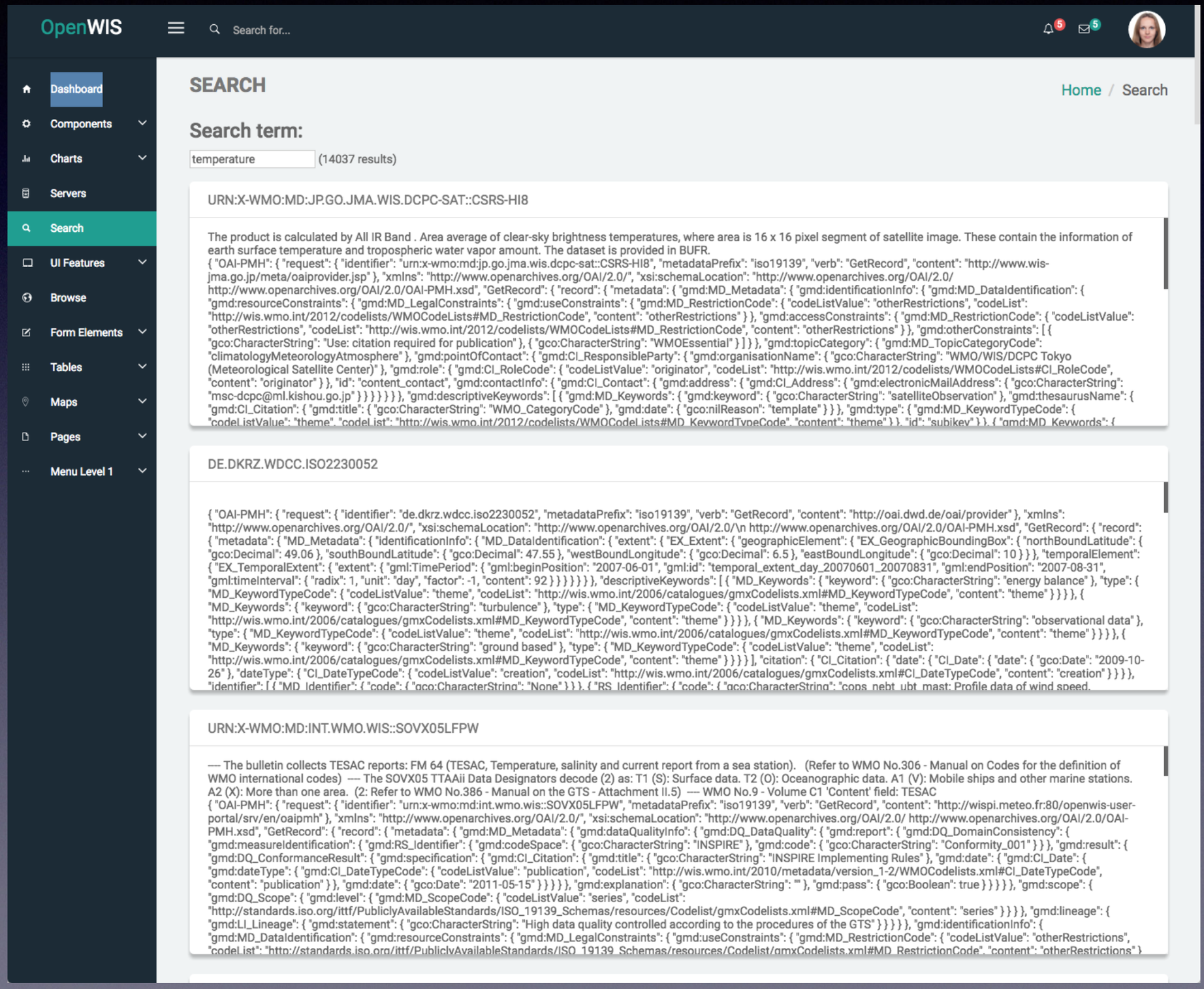
PoC - Adding servers for harvesting



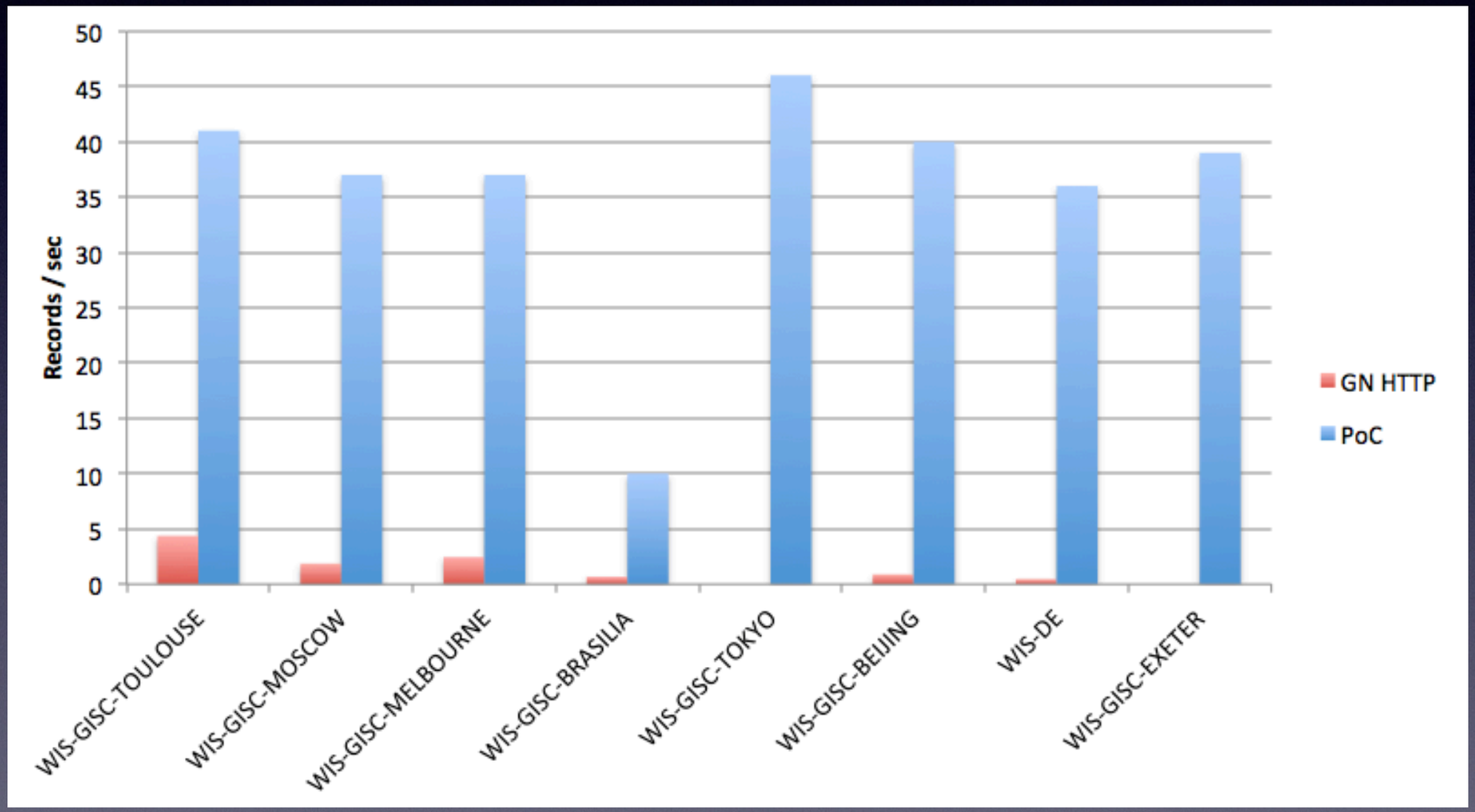
OpenWIS v5
PoC - Browsing results



OpenWIS v5
PoC - Searching with Elasticsearch



OpenWIS v5
PoC - Benchmarks



OpenWIS v5

PoC - Try it out

```
curl https://raw.githubusercontent.com/NMichas/OpenWIS-PoC/master/docker-compose.yml | docker-compose -f - up
```

If your Docker Engine is behind a proxy, you can inject your proxy's info as:

```
curl https://raw.githubusercontent.com/NMichas/OpenWIS-PoC/master/docker-compose.yml | HTTP_PROXY=http://proxy:port docker-compose -f - up
```