



Green mobility data models and services for smart ecosystems

D4.2 GreenMov Reference Architecture and guidelines v2

Document Identification	
Contractual Delivery Date	28/02/2023
Actual Delivery Date	28/02/2023
Responsible Beneficiary	FIWARE Foundation
Contributing Beneficiaries	IMEC, ATOS
Dissemination Level	PU
Version	1.0
Total Number of Pages:	61

Keywords
Sustainable Mobility, Data Models, Reference Architecture



This document is issued within the frame and for the purpose of the GreenMov project. This project has received funding from the European Union's Innovation and Networks Executive Agency – Connecting Europe Facility (CEF) under Grant AGREEMENT No INEA/CEF/ICT/A2020/2373380 Action No: 2020-EU-IA-0281. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

This document and its content are the property of the *GreenMov* Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the *GreenMov* Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the *GreenMov* Partners.

Each GreenMov Partner may use this document in conformity with the GreenMov Consortium Grant Agreement provisions

(*) Dissemination level.-PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

Related Activity	Activity 4	Document Reference	D4.2
Related Deliverable(s)	D4.1	Dissemination Level (*)	PU

List of Contributors	
Name	Partner
Alberto Abella	FIWARE Foundation
Brecht Van de Vyvere	IMEC
Rémi Ollivier	ATOS

Document History			
Version	Date	Change editors	Changes
0.1	21/09/2022	Alberto Abella (FF)	Starting ToC
0.2	14/11/2022	Alberto Abella (FF)	Contributions from Activity 5
0.3	28/11/2022	Brecht Van de Vyvere (IMEC)	Update coverage index, explain source selection library
0.4	01/12/2022	Alberto Abella (FF)	Reordering and filtering contents
0.5	21/12/2022	Alberto Abella (FF)	Indexing illustrations and tables. Review of the content
0.53	31/01/2023	Alberto Abella (FF)	Restructuring because of creation of deliverable 4.3.
0.6	08/02/2023	Alberto Abella (FF) Julian Andrés Rojas (IMEC)	Extension of the LDES description and completed the components installation description
0.65	15/02/2023	Remi Ollivier (ATOS)	Contribution to section 2.4

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	2 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Document History			
Version	Date	Change editors	Changes
0.7	16//02/2023	Alberto Abella (FF)	References review and new ones
0.8	22//02/2023	Alberto Abella(FF)	Review structure and contents based on the comments of the internal reviewer
0.9	27//02/2023	Alberto Abella(FF)	More restructuring, adding details and formatting
0.91	27//02/2023	Alberto Abella(FF)	Removed tracked changes
0.92	27//02/2023	Alberto Abella(FF)	Formatting
0.95	28/02/2023	María Guadalupe Rodríguez (ATOS)	Quality Review Form
1.0	28/02/2023	Carmen Perea (ATOS)	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who (Partner short name)	Approval Date
Reviewer	Ignacio Elicegui (ATOS)	27/02/2023
Quality manager	María Guadalupe Rodríguez (ATOS)	28/02/2023
Project Coordinator	Carmen Perea (ATOS)	28/02/2023

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	3 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Table of Contents

Document Information	2
Table of Contents	4
List of Figures	8
List of Acronyms	9
Executive Summary.....	10
1 Introduction	11
1.1 Purpose of the document.....	11
1.2 Relation to other project work.....	11
1.3 Structure of the document	11
1.4 Glossary adopted in this document	12
2 Basic software components	14
2.1 Introduction to FIWARE architecture, standards and components.....	14
2.2 Reference architecture levels	14
2.3 Global diagram.....	15
2.4 Core components of the reference architecture.....	18
2.4.1 Context broker	18
2.4.2 NGSI-LDES	18
2.4.3 IoT Agents	19
2.5 Persistence components of the reference architecture.....	20
2.5.1 Cygnus-LD.....	20
2.5.2 Other persistence components	20
2.6 Security components	20
2.6.1 Keycloak	20
2.6.2 Keyrock.....	21
2.6.3 Wilma.....	21
2.6.4 Authzforce	21
2.7 Other components	21

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	4 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

2.7.1	Connection with open data portals.....	21
3	Deployment of the platform	23
3.1	Installation of the core context broker	24
3.1.1	Installation of Orion-LD on MongoDB	24
3.2	Persistence Components	24
3.2.1	Installation of Cygnus	24
3.3	Installation of security components	25
3.3.1	Installation of Keycloak.....	25
3.3.2	Installation of Keyrock	25
3.3.3	Installation of Wilma	25
3.3.4	Installation of Authzforce	26
3.4	Configuration of federated scenarios	26
3.4.1	Types of deployments simple and advanced.....	26
3.4.2	Advanced deployments.....	26
3.4.3	Federated deployments	26
3.4.4	Multitenancy	27
3.4.5	Distributed operation modes	28
4	Operation of the platform	30
4.1	Operational aspects	30
4.2	Secure code, from design to delivery	30
4.3	Secure by design	30
4.4	Dependencies scanning	31
4.5	DevSecOps.....	31
5	Data Architecture	34
5.1	Data Storage architecture and technical format	34
5.2	Basic data classes / entities	35
6	Conclusions	37
7	References	38
	Annex I. Requirements for a generic enabler.....	40

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	5 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Licensing and open SSF Best practices signature.....	40
General requirements	41
Documentation requirements	41
Development requirements	41
Annex II. An example of configuration.....	42
How to setup	42
1. Prepare AWS account	42
2. Install OpenShift cluster	42
3. Install certificates	43
Clone the acme.sh github-repo	43
Setup AWS credentials.....	43
Obtain certificates	43
Create the secrets.....	43
Patch ingress-controller and api-server	43
Update kubeconfig	44
Verify success	44
4. Install ArgoCD.....	44
Create namespace.....	44
Install ArgoCD operator.....	44
Deploy an instance of ArgoCD	45
5. Prepare ArgoCD for namespaced deployments	46
6. Deploy namespaces.....	47
Click create and wait until its running:	49
7. Deploy bitnami/sealed-secrets	49
Click create and wait until its running:	51
8. Create secrets	51
Install kubeseal.....	52
Seal the secret.....	52
Push the mongodb-sealed-secret.yaml file to your repository	53

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	6 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Fill out the form - in contrast to "sealed-secrets" this will consist of plain manifests(like "namespaces")	53
Click create and wait until the sealed-secret is deployed and an unsealed secret is created from it:	54
9. Deploy MongoDB.....	54
Click on "NEW APP"	54
Fill out the Form.....	54
Click create and wait :.....	55
10. Deploy Orion-LD.....	56
Click on "NEW APP"	56
Fill out the Form.....	56
Advanced topics.....	56
Annex III. Concepts.....	57
Reactive manifesto.....	57
Blue-Green deployment	57
Canary deployment	58
Annex IV. Security.....	59
General security considerations	59
Security of communications	60
Management of secrets	60
Slow down attackers	61
Intrusion detection system	61
Data integrity	61

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	7 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

List of Figures

Figure 1: Global diagram for the reference architecture. source: FIWARE reference architecture for cities adapted for GreenMov.....	18
Figure 2: Scenario for broker federation	28
Figure 3: Diagram for distributed operation modes	29
Figure 4: Entity and attributes	35
Figure 5: Argo console installation	43
Figure 6: Argo console configuration.....	43
Figure 7: Argo deployment screen	44
Figure 8: Argo namespaces configuration.....	45
Figure 9: Argo. Creation of new app.....	46
Figure 10: Argo. Configuration of new app	46
Figure 11: Argo. Monitoring of app	47
Figure 12: Argo. Create new app	47
Figure 13: Argo. Creation of new app.....	48
Figure 14: Argo. Monitoring of apps.....	49
Figure 15: Argo. Creation of new app.....	52
Figure 16: Argo. Secret configured	52
Figure 17: Argo. New app	53
Figure 18: Argo. Deployment of Mongodb.....	54
Figure 19: Argo. Creation of new app.....	54

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	8 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

List of Acronyms

Abbreviation / acronym	Description
CI/CD	Continuous integration / continuous deployment. Set of practices used in software development that aim to streamline the development and deployment process, making it faster and more efficient.
CSR	Context Source Registrations
Dx.y	Deliverable number y belonging to Activity x
EC	European Commission
GE	Generic Enabler. Every approved component of the FIWARE framework.
JSON	JavaScript serialized object notation. It is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values)
JSON schema	JSON Schema specifies a JSON-based format to define the structure of JSON data for validation, documentation, and interaction control.
JSON-LD	It is a method of encoding linked data using JSON.
LDES	Linked Data Event Stream
ODALA	ODALA is an initiative (European project) that aims to promote the use of Big Data to facilitate and speed up decision-making in public administrations. This initiative has a social focus and is designed to help the use of Smart Cities technology simply and practically.
SHACL	Shapes Constraint Language. Language for describing Resource Description Framework graphs.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	9 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Executive Summary

This document outlines the second version of the GreenMov common reference architecture aimed at implementing city use cases. It extends the security components described in the first version and remove others not useful for the current use cases. It also incorporates approaches for enhancing scalability and provides a set of guidelines for specific scenarios using city pilot examples. It also incorporates feedback gathered from the use cases.

The architecture supports smart city services under activity 3 and the implementation of the use cases under activity 5, as well as partially integrates data sources and entities stored in different elements of the architecture described in the data models described in activity 2.

It is mostly based on FIWARE enablers although other components, Grafana, are also included. It also provides links for deploying necessary components to support local customization in each use case. This deliverable is complemented by document D4.3 which delves into advanced concepts, such as performance analysis of core components in various configurations and configuration of a system federation. The document outlines the high-level architecture and its elements and how they connect, covers data persistence in services, and provides identification and access control. It ends with recommendations on platform operation and complementary information on main element configurations in the annexes.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	10 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

1 Introduction

1.1 Purpose of the document

This document extends the content of the deliverable 4.1 with a tighter integration of the requirements coming from the use cases and adding more specific instructions on how to deploy the reference architecture.

1.2 Relation to other project work

This document is the second version of the GreenMov reference architecture, including new components and an extended description of the software components and data flows. Additionally, the performance analysis and other configuration options have been also extended from previous deliverable 4.1. It is also added some consideration for the deployment and exploitation and about security.

1.3 Structure of the document

This document is structured in 6 major chapters, including this one **Chapter** Introduction 1 and 3 annexes.

The **chapter 1** is an introduction to the purpose of this document, its relationship to other documents generated, its overall structure and the glossary of terms to understand its full content of the document.

Chapter 2 describes the main components to be used from a high-level architecture diagram to the detailed description of the individual components and their connection with other external systems. It includes an analysis of the performance of the core component, the context broker, in different configurations and implementations.

Chapter 3 provides guidance on the deployment of the different components, including the data persistence components, those related with authentication and access control, visualization and how to federate different instances.

Chapter 4 introduces operation aspects to run the platform and includes a specific section regarding security.

Chapter 5 states the principles of the data architecture that is further developed in the documents generated of the activity 2.

Chapter 6 presents the conclusions of the document.

There are also 4 **annexes**, the first one devoted to the description of the requirements of a software package to become a generic enabler approved to belong to the FIWARE framework. The second annex describes an extensive example on how to deploy several of the core components of the platform. Finally, the third annex describes 3 key concepts for the operation of the platform and the fourth provides some recommendations for the security of the platform.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	11 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

1.4 Glossary adopted in this document

This glossary replicates most of the terms of the glossary in deliverable 4.1 and it is included here only for clarification purposes for those readers that have not read previous deliverable.

- **Context broker.** A Context Broker is an open-source platform that allows developers to manage and store contextual information about a network of smart objects, devices and other sources of data. It provides a unified view of the contextual information and allows for real-time querying, geoquerying and updates of the information. The Context Broker can be used in IoT, smart city and smart industry applications to improve data management and enable better decision-making. The main functions of a context broker are:
 - Context information storage: The context broker stores context information, such as device data or application data, in a centralized repository.
 - Context information retrieval: The context broker enables applications and devices to retrieve context information as needed.
 - Context information distribution: The context broker can distribute context information to multiple applications and devices in real-time.
 - Context information management: The context broker provides APIs for managing context information, such as creating, updating, and deleting entities.
 - Context information integration: The context broker can integrate context information from multiple sources, such as sensors, devices, and applications, into a unified view.
 - Context information querying: The context broker enables applications to query context information in a flexible manner, such as by location, time, or entity type.
 - Context information subscription: The context broker can support subscriptions, allowing applications to receive notifications when the context information they are interested in changes like reaching a threshold or the creation of an attribute.
- **Cygnus.** It is a software package for managing the history of the context that is created as a stream of data which can be injected into multiple data sinks, including some popular databases like PostgreSQL, MySQL, MongoDB or AWS DynamoDB as well as BigData platforms like Hadoop, Storm, Spark or Flink.
- **Draco** is another software package to provide a data persistence mechanism for managing the history of context. It is based on Apache NiFi and is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic and also offers an intuitive graphical interface.
- **Kubernetes.** Kubernetes, often abbreviated as "K8s", is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Containers are a lightweight and portable way to package and deploy applications, and Kubernetes provides a way to manage and automate these containers at scale. It can run on a variety of platforms, including public, private, and hybrid clouds, as well as on-premises data centers.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	12 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- **Kurento.** is a component that enables real-time processing of media streams supporting the transformation of video cameras into sensors as well as the incorporation of advanced application functions (integrated audiovisual communications, augmented reality, flexible media playing and recording, etc).
- **Linked data.** It is structured data, which is interlinked with other data, so it becomes more useful through semantic queries.
- **Man-in-the-middle.** Cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other
- **Near to real-time system.** A near to real-time system is a information system that processes and responds to inputs in a timely manner, where the response time is quite relevant but not critical for the correct functioning of the system. For this document it means in the range of a second to few minutes.
- **Orion.** Software solution for context information management compliant with NGSIv2 specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [1].
- **Orion-LD.** Software solution for context information management compliant with NGSI-LD specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [2].
- **Real-time system.** A real-time system is a information system that processes and responds to inputs in a timely manner, where the response time is critical for the correct functioning of the system. For this document it means in the range of fraction of a second or shorter. In real-time systems, the response time is strictly bounded and determined by the system's requirements, and the correctness of the system's behavior depends not only on the logical result of computations, but also on the time at which the results are produced.
- **Scorpio.** Software solution for context information management compliant with NGSI-LD specification, created by NEC and other entities, available as a Generic enabler of the FIWARE platform [3].
- **Stellio.** Software solution for context information management compliant with NGSI-LD specification, created by EGM and other entities, available as a Generic enabler of the FIWARE platform [4].

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	13 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

2 Basic software components

2.1 Introduction to FIWARE architecture, standards and components

FIWARE NGSI is the API exported by a FIWARE Context Broker, used for the integration of platform components within a "Powered by FIWARE" platform and by applications to update or consume context information. FIWARE NGSI API specifications have evolved over time and now it is a standard under the umbrella of ETSI ISG CIM group [5] which the name ETSI NGSI-LD standard. The FIWARE Community plays an active role in the evolution of ETSI NGSI-LD specifications which were based on NGSIv2 and commits to deliver compatible open-source implementations of the specs. The FIWARE Community plays an active role in the evolution of ETSI NGSI-LD specifications and commits to deliver compatible open-source implementations of the specs (e.g., Orion-LD, Scorpio, and Stellio).

Building around the FIWARE Context Broker, a rich suite of complementary FIWARE Generic Enablers are available, dealing with the following functionalities:

- Core Context Management manipulates and stores context data so it can be used for further processing. Dark blue rectangle in the middle for the diagram with the entities stored in it and some of their attributes.
- Interfacing with the Internet of Things (IoT), Robots and third-party systems, for capturing updates on context information and translating required actuation. Turquoise sections and light blue ones in the lower part of the diagram.
- Processing, analysis and visualization of context information, implementing the expected smart behavior of applications and/or assisting end users in making smart decisions. Upper part with red background and boxes below them.
- Interfaces with other systems and identify and access management on the right part of the diagram.

2.2 Reference architecture levels

The Reference Architecture should address how the integration of data and information across different systems on the use cases will be achieved, ensuring sustainable and efficient service provisioning. Interoperability should be supported in line with relevant standards (REST, NGSI, JSON-LD).

FIWARE is a curated framework of open-source platform components which can be assembled together and with other third-party platform components to accelerate the development of Smart Solutions. The main and only mandatory component of any "Powered by FIWARE" platform or solution is a FIWARE Context Broker Generic Enabler, bringing a cornerstone function in any smart solution with the purpose to manage the context information created from one or different context providers and consumed by one or several context consumers.

In this respect, the Context Broker technology will play a cornerstone role for the integration, following a system of systems approach, of data and information across the systems implementing the different services

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	14 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

for a given focus area (e.g., traffic management, environmental impact, or other system in relation to public services). Some of these systems will be provided globally as a service from public clouds. The Context Broker building block has been the preferred technology for other EU programs.

The figure in the next chapter shows a diagram with the main components based on a FIWARE reference architecture for smart cities, where the requirements for the different use cases Murcia y Molina, Nize and Flanders has been taken into consideration.

The central element of this architecture is the context broker where the retrieval and sharing of the information happens.

It has four main levels. The upper level is the one for the connection with the legacy systems of the city or other general applications. This level will be extremely customized for every use case.

The second level is the one with the context broker and some adapters for the connection with the legacy systems. It has to be noted that in right side of this level there are the identification and identity blocks. The use and sharing of these elements in the second level allow the flexibility to provide solution to the different use cases but keeping the same basic software structure which allow scalability and shared knowledge. Besides this, the fact that these components come from a FIWARE reference architecture ensure a wide knowledge base to solve any issue of the use cases.

The third level of the schema is populated by the adapters with the IoT world, cameras, sensors, etc. Again, being based on the FIWARE components ecosystem ensures the availability of many solutions for the connection a large database of integration with other IoT elements and some share software frameworks for these cases in which a customization is required.

The lower level (fourth) includes those IoT elements and other data sources for the solution of the city that can be connected thanks to the previous level without much hassle. This level retrieve data from different sources that can be modeled into common data models, eventually allowing seamless data sharing between use cases or city's ecosystems and being stored into entities in the second level.

2.3 Global diagram

This simple diagram was created after gathering the requirements from the use cases in Activity 5 and adapting a proven reference architecture designed by the FIWARE community for smart cities.

The main elements depicted in the next diagram are:

- A Context Broker component [6], is at the core of the architecture, keeping a digital twin representation of the real-world objects and concepts relevant to the specific problem tackled: Environmental sensors, traffic sensors, noise sensor, bike stations, etc.
- Southbound to the Context Broker, the NGSI IoT Agents, available as part of the FIWARE IDAS framework, are used for connections to the different sensors or actuators, used for example to detect available bikes on the stations. They perform the necessary conversions between IoT protocols and NGSI. In addition, System Adapters developed based on the IDAS Agent library cope with the connection to the legacy systems of the city as described in the services architecture. FIWARE

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	15 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

component Kurento is able to process the video streams of cameras deployed in the shop floor, which are helpful to detect potential obstacles or risky situations.

- Southbound to the Context Broker, data sources can be serialized in multiple formats (JSON, CSV , JSON-LD...) and published through heterogeneous APIs (JSON API, LDES, NGSI-LD).

A combination of open-source components from third party products and advanced data maps for monitoring processes. A number of FIWARE Data Connectors (Cygnum, Draco, Cosmos, STH Comet, QuantumLeap) are available as part of FIWARE to facilitate transference of historic context / digital twin information to these tools.

- Transversal to all these layers, a number of FIWARE components support Identity and Access Management (e.g., Keycloak, Keyrock). They control the flow of data across the different layers. With regards to the access to the Context Broker, they enforce the policies establishing what users can update, query or subscribe to changes on context / digital twin data. Note that the flow of data is not only south to north in the picture. Northbound applications can perform updates on context data, which in turn will trigger changes in the sensors, actuators or systems that are connected southbound.
- Northbound to the Context Broker, there are several tools aimed at supporting real-time processing of the historical data streams generated as context/digital twin information evolves over time. Additionally, the NGSI-LDES component creates a scalable interface for data sharing. The component allows applications to replicate and synchronize with the historic and real-time context of entities. Example of such an application is the Coverage Index or Registry API, which can be served by a Context Registry and can be used by a source selection component. NGSI-LDES requires that the temporal and types of interfaces are available on the Context Broker.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	16 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

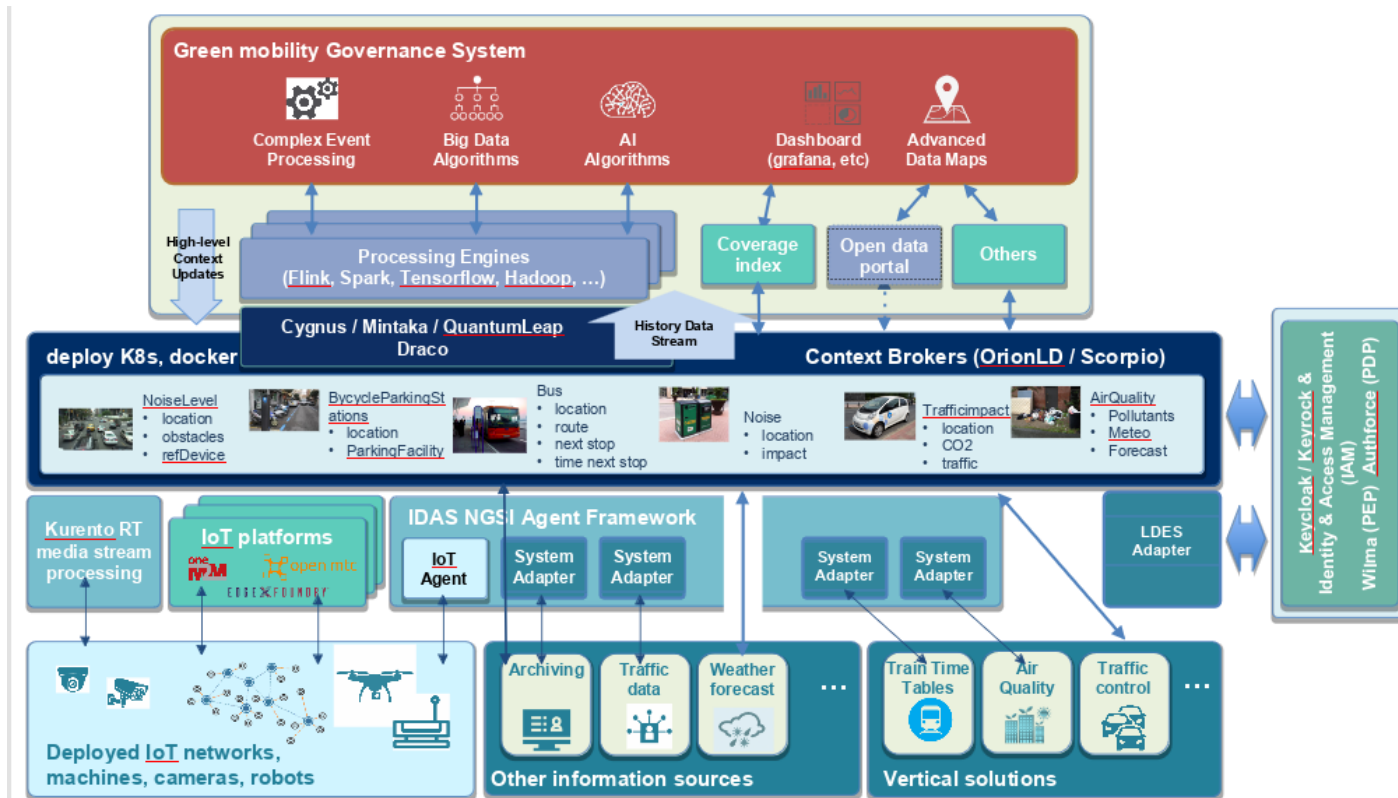


Figure 1: Global diagram for the reference architecture. source: FIWARE reference architecture for cities adapted for GreenMov.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	17 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

2.4 Core components of the reference architecture

2.4.1 Context broker

The context broker integrates information from sensors, systems and other machines in the different use cases breaking information silos. It allows not only the retrieval of information from heterogeneous sources but also the querying in time and the geoquerying in spatial dimension. Additionally, it also allows the users to subscribe to changes or updates and to receive notifications when the conditions are met.

This broker must support following entry points to be compatible with the NGSI-LDES:

- the NGSI-LD temporal interface.
- the NGSI-LD types interface.

Examples of Context Broker we see fit: Orion-LD + Mintaka, Scorpio, Stellio.

To be able to choose between these brokers, some load tests have been made to compare the brokers performance depending on the number of entities in the broker.

Examples of Context Broker we see fit: Orion-LD + Mintaka, Scorpio, Stellio.

- Orion-LD is an NGSI-LD compliant context broker part of FIWARE. It is developed in C++, and it runs under Linux. Orion-LD uses MongoDB, a document-oriented database, as a context database. Orion-LD also implements Temporal Representation of Entities with Timescale a time-series SQL database. Mintaka is the component used to manage temporal queries.
- Scorpio is an NGSI-LD compliant context broker developed by NEC Laboratories Europe and NEC Technologies India. Scorpio is developed in Java but there are two different versions of Scorpio, one is built with Spring Boot and another one with Quarkus. Both versions are built with Apache Maven. Scorpio works with Apache Kafka as message bus and PostgreSQL with PostGisextension to store context and historic data. Scorpio allows to create distributed and federated deployment of various context brokers.
- Stellio is an NGSI-LD compliant context broker developed by EGM. It is developed in Kotlin with Spring Boot framework, and it is built with Gradle. Stellio is composed of three main services: search service, subscription service and API gateway. It uses a Kafka component as message bus and a TimescaleDB to store context and historic data.

To be able to choose between these brokers, some load tests have been made to compare the brokers performance depending on the number of entities in the broker.

The deliverable 4.3 contain the advance criteria to make the decision depending on the needs of the use cases.

2.4.2 NGSI-LDES

The NGSI-LDES component acts as a scalable data synchronization/exchange interface that publishes one or more Linked Data Event Streams (LDES) from NGSI-LD compliant context brokers. LDES defines a hypermedia-based Web API that provides access to immutable and semantically annotated data subsets

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	18 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

containing the (historical) state of a certain data collection. For example, it can be used to read the latest state of a bicycle sharing station but also allows to query for previous states of that same station, as reported in the past.

To allow for efficient and scalable data access, an LDES defines a certain logical data structure that ensures that historical data records, which will not change any further, are served as immutable resources (in terms of HTTP caching), lowering the cost for serving historical data and thus improving scalability. Multiple logical data structures can be implemented for an LDES, (e.g., linked lists, b-trees, skip lists, etc.). The choice is made at design time and largely depends on the underlying data sources and their querying capabilities. In the context of GreenMov, a b-tree like hierarchical data structure was chosen, which is supported by the temporal querying interfaces of NGSI-LD context brokers. Each node in the tree represents aggregated time windows of a certain granularity (e.g., week, day, hour, etc.). Such design allows the LDES to behave as a virtualized view over the data that is hosted in a context broker, i.e., the data does not need to be duplicated anywhere else, and also lowers the cost for historical queries, since historical fragments only need to be request to the context broker once and be served from cache onward.

In terms of data content, the NGSI-LDES module produces an independent LDES stream for every entity type that can be found within a NGSI-LD context broker. Each LDES will continuously produce versioned members (as in LDES/TREE notation) which will contain links to the respective (versioned) entities which are defined in correspondence to the specific (smart) data model used by the context broker.

Additionally, the NGSI-LDES component leverages the NGSI-LD types of interfaces of a context broker to automatically generate a compliant DCAT metadata catalog.

NGSI-LDES is available as open source on GitHub [7].

2.4.3 IoT Agents

The IoT agents gather information from specific IoT sensors and transfer them into the main components of the platform. These are identified as potentially usable across the platform depending not only on the existing sensor but on other that could be included during execution of the project and beyond.

- IoT Agent for JSON [8] - a bridge between HTTP/MQTT messaging (with a JSON payload) and NGSI/NGSI-LD.
- IoT Agent for LWM2M [9] a bridge between the Lightweight M2M protocol and NGSI/NGSI-LD
- IoT Agent for Ultralight [10] - a bridge between HTTP/MQTT messaging (with an UltraLight2.0 payload) and NGSI/NGSI-LD
- IoT Agent for LoRaWAN [11] - a bridge between the LoRaWAN protocol and NGSI/NGSI-LD
- IoT Agent for Sigfox [12] - a bridge between the Sigfox protocol and NGSI/NGSI-LD
- IoT Agent Library [13] - library for developing your own IoT Agent, almost all the IoT Agents are using this library to develop their concrete bridge between legacy systems and NGSI/NGSI-LD.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	19 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

2.5 Persistence components of the reference architecture

2.5.1 Cygnus-LD

The Cygnus-LD [14] Generic Enabler enables the persistence of historical context data through the creation of data streams and can be injected into multiple data sinks, including many popular databases such as PostgreSQL, ArcGIS or public Open Data Platform like CKAN. Cygnus is based on Apache Flume. Potentially required when using Orion-LD because persistence in Scorpio is already available.

- **Data management:** Cygnus-LD allows you to store and manage Linked Data in a way that is consistent with the Linked Data principles.
- **Data integration:** Cygnus-LD provides tools for integrating Linked Data from different sources, such as databases and web services. This allows you to create a unified view of your data and to link related information from different sources.
- **Data interoperability:** Cygnus-LD supports the Linked Data standards and protocols, allowing you to share data with other systems and applications that use Linked Data. This enables interoperability between different systems and makes it easier to build applications that use Linked Data.
- **Data privacy and security:** Cygnus-LD provides tools for managing access to Linked Data, allowing you to control who can view and modify your data. This helps to ensure the privacy and security of your data.

2.5.2 Other persistence components

Although there are other persistence components for persistence in the FIWARE framework, Quantum Leap [15], Cosmos [16], there are not initially used by the use cases and therefore not included here.

2.6 Security components

2.6.1 Keycloak

Keycloak is an open-source identity and access management solution. It helps organizations to secure their applications and services by providing a single point of access for authentication and authorization. Keycloak offers features such as user management, multi-factor authentication, and support for social login providers. It also integrates with other identity providers such as LDAP and Active Directory. Keycloak is written in Java and is available under the Apache License 2.0.

It has these features:

User management: Keycloak provides a centralized user management system, where you can manage users and their permissions. This includes features such as user registration, password policies, and account recovery.

- **Multi-factor authentication:** Keycloak supports multiple authentication methods, such as OTPs, biometrics, and security questions. This allows you to set up a secure authentication system for your applications and services.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	20 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- **Social login:** Keycloak integrates with popular social login providers, such as Facebook and Google, allowing users to log in to your applications using their social media accounts.
- **Identity brokering:** Keycloak can act as an identity broker, allowing users to log in to your applications using credentials from other identity providers, such as LDAP or Active Directory.
- **Single sign-on (SSO):** Keycloak supports SSO, allowing users to log in to multiple applications with a single set of credentials. This makes it easier for users to access the applications they need, and it helps to improve security by reducing the number of passwords that users need to remember.

2.6.2 Keyrock

The Keyrock [16] Generic Enabler acts as the Identity Management component and provides secure and private authentication, basic authorization, and identity federation for applications. It plays a crucial role in ensuring security within the FIWARE System of Systems architecture. The Keyrock component includes tools for administrators to manage user life cycle functions. It complements some aspects of the previous Keycloak component and serves as an alternative in others.

2.6.3 Wilma

Wilma serves as the standard implementation of a PDP due to its complete integration with the FIWARE ecosystem. It is designed to operate seamlessly with OAuth2 and XACML protocols, the authentication and authorization standards adopted by FIWARE. Moreover, every GE incorporates this component on top of their REST APIs, making it widely tested and utilized in diverse scenarios.

2.6.4 Authzforce

Authzforce is the GE that provides with the reference implementation of the Authorization PDP Generic Enabler (previously known as Access Control GE). As per the GE specification, this implementation offers an API that enables you to obtain authorization decisions based on authorization policies and requests from PEPs. The API adheres to the REST architecture style and conforms to XACML v3.0. XACML, which stands for eXtensible Access Control Markup Language, is an OASIS standard used for authorization policy format and evaluation logic, as well as for the request/response format for authorization decisions. The XACML standard defines the terms PDP (Policy Decision Point) and PEP (Policy Enforcement Point). This GE reference implementation acts as a PDP.

To comply with the XACML architecture, you may require a PEP (Policy Enforcement Point) to safeguard your application, which is not included in this implementation.

2.7 Other components

2.7.1 Connection with open data portals

This is a feature under discussion in the use cases. Anyhow information about the connector to transfer the data from the services to open data portals based on CKAN is included.

The CKAN Extension. Publishing and consuming open data is a keystone for the development of applications and the creation of an innovation ecosystem. CKAN is one of the most extended Open Data publication

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	21 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

platforms and is becoming the de-facto standard for data publication in Europe. Moreover, CKAN is an open-source platform which means it can be easily adapted and expanded and integrated into multiple use cases.

The CKAN Extension [18] integrates CKAN solution with the FIWARE platform, enabling the right-time context information served by a FIWARE Context Broker and to be published as a dataset resource, making it easier to be discovered and consumed as Open Data content. Additionally, this extension allows the integration with FIWARE Security in order to enrich the access control and enable explicit acceptance of data terms and conditions, usage accounting, or data monetization.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	22 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

3 Deployment of the platform

The deployment infrastructure must obey to some requirements:

- Be easy to deploy by the technical teams of the pilot sites on their infrastructure.
- Be responsive as the platform deals with near to real time data management and processing and users needs immediate feedback for decision making.
- Be highly available as the platform deals with near to real time data management and processing and users need feedback at any time for decision taking.
- Be scalable as more in more data and usages will come in and the platform must stay responsive and with the right performance over time.

Such concerns are deeply tied to the global architecture of the platform and the components that are integrated. In this respect, it is very important that the platform and its components adhere totally to the Reactive Manifesto (See Annex III. Concepts), which defines the core principles that must be followed by any modern reactive architecture.

Then, it has to be backed by a deployment platform that will bring the ease of deployment, and the tools to allow for high availability and scalability.

Nowadays, Kubernetes is de-facto standard for such deployments:

- Deployments can be formalized and automatized, especially via the use of Helm charts.
- Integrated support for load balancing.
- Integrated support for horizontal scaling.
- Automatic restart of containers when a node dies or when a container does not respond to health checks.
- Automatic placement of containers based on their requirements.

Furthermore, it brings another important feature, by offering to progressively roll out changes to a deployed platform in production, while monitoring application health to ensure all the services are still up for end users. If something goes wrong, Kubernetes will roll back the changes (whether automatically or manually). This allows for advanced deployment strategies like Blue-Green deployment (See Blue-Green deployment), Canary deployments (Set Canary deployment), and so on.

Finally, it is expected a tight integration between the CI/CD tool and the deployment platforms (whether production, integration, development, ...). There exist tools (like JenkinsX) that permit such a seamless experience, they will be considered first (as long as well-known tools in this domain, like TravisCI or Bamboo).

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	23 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

3.1 Installation of the core context broker

3.1.1 Installation of Orion-LD on MongoDB

There is an extensive documentation on how to install Orion-LD on MongoDB. The installation instructions are available for the base platforms [19]:

- use a prebuilt docker image, or

build Orion-LD from source code:

- Ubuntu 18.04.3 LTS - the Official Distribution
- Ubuntu 20.04.1 LTS
- Ubuntu 22.04 LTS - no official instruction from MongoDB on how to install their DB

Installation of Scorpio on PostgreSQL

The installations instruction [20] recommends its installation based on docker compose, although it can also be installed based on the source code. The instructions also include a tutorial on how to install PostgreSQL (version 10).

3.2 Persistence Components

3.2.1 Installation of Cygnus

FIWARE Cygnus is an open-source component of the FIWARE platform for the Internet of Things (IoT). It is a connector that allows for the storage and management of context information from IoT devices and sensors in different backends, such as databases, cloud storage systems, and data warehouses. Cygnus acts as a bridge between the IoT devices and the backend storage systems, enabling seamless and efficient data transfer and management. By using Cygnus, developers can easily store and manage context information from IoT devices, enabling the creation of advanced IoT applications.

The installation of Cygnus demands a previous analysis of the different types of data sources to be ingested in the component. A large catalog of them is available including (not exhaustive list). The installation guide is available in its main page [21].

- Integration with REST interfaces
- Databases
 - CartoDB
 - DynamoDB
 - ElasticSearch
 - MongoDB
 - MySQL
 - Oracle
 - PostgreSQL

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	24 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- Big data resources
 - Kafka
 - HDFS
- Other FIWARE components
 - Orion
- Geographical systems
 - Arcgis
 - PostGIS

3.3 Installation of security components

3.3.1 Installation of Keycloak

Like most of the components it can be installed based on containers like dockers and Kubernetes. It requires the installation of JDK (i.e. OpenJDK) either docker or Kubernetes. and eventually Openshift when using Kubernetes.

The official guide for the installation of Keycloak can be found in this reference [22].

3.3.2 Installation of Keyrock

In order to be able to run Keyrock, it is needed to have previously installed the following software components:

- Node.js and Node Packaged Modules. They are usually included within Node.js.
- MySQL

The standalone configuration requires to configure the port for providing service, the connection with the database and an initial population of the database. It also requires creating the session and encryption password.

It is also possible to be installed by using a docker image [23].

The official guide for the installation of Keyrock can be found in this reference [24]

3.3.3 Installation of Wilma

Wilma requires to install these elements:

- Node.js >= V8.x.x
- npm >= 5.x.x.

It can be installed from the source code or using a docker image.

The official installation guide is found in this reference [25].

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	25 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

3.3.4 Installation of Authzforce

Authzforce is based on Java and therefore needs to get in the environment:

- JRE 11 from OpenJDK
- Tomcat 9.X

It requires a special configuration of Tomcat, additionally other configuration parameters are required for its successful installation. The official installation guide can be found in this reference [26].

3.4 Configuration of federated scenarios

3.4.1 Types of deployments simple and advanced

Federation of brokers is a feature specially developed in the version 1.6.1 of NGSI-LD standard (Aug.2022). This version also addresses some other evolution like the concise format, the representation of deleted entities and attributes in notifications, the temporal evolution and some conventions for using NGSI-LD for actuation (not only to gather information and answering about it but also to make other systems to act).

Usually, the simple deployments typically are controlled by a single operator that can guarantee consistency. The size of the system only comprises some hundreds of entities. Also typically are used by a single application or a small set of applications and it can be handled by a single Context Broker / single database.

However, in order to provide services to a full city this configuration is limited and therefore the advance deployments has to include the scalability for dealing with thousands or even millions of entities. It needs to be allowed the interaction of multiple independent operators. Besides this, they can have heterogeneous sources of overlapping information. In terms of access permission, it has to be ready to require that only some of the information is shared, and of course there will be many different applications with different access level permissions. In such deployments it is required that multiple context brokers interact with each other and the answer to the queries will need to gather information from multiples instances and to recombine globally.

3.4.2 Advanced deployments

NGSI-LD is a standard that enables distributed deployments by supporting multiple context sources. These sources can include full context brokers, IoT agents, or other entities that partially implement the NGSI-LD API. The context brokers can register with the context registry to specify the information they can provide. The context broker then accesses and aggregates the information from the context registry to return it to the requesting application. Furthermore, context brokers can themselves have other context brokers registered, allowing for the creation of hierarchical structures that reflect company or geographical structures.

3.4.3 Federated deployments

A federated deployment is a type of distributed deployment that allows applications to access context information from different sources, each running its own context broker. It is not technically different from other distributed scenarios, but the key difference is that it operates across administrative boundaries and lacks central control. In federated scenarios, the primary focus is typically on accessing and aggregating context

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	26 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

information from multiple context brokers, while the management of the information, such as creation, update, and deletion, is performed locally within each domain.

In order to make compatible data for the same entity there is a new built-in sub attribute (`datasetId`) which depending on the source could help to make the difference. Additionally, scopes can help on filtering information coming from a group of sources.

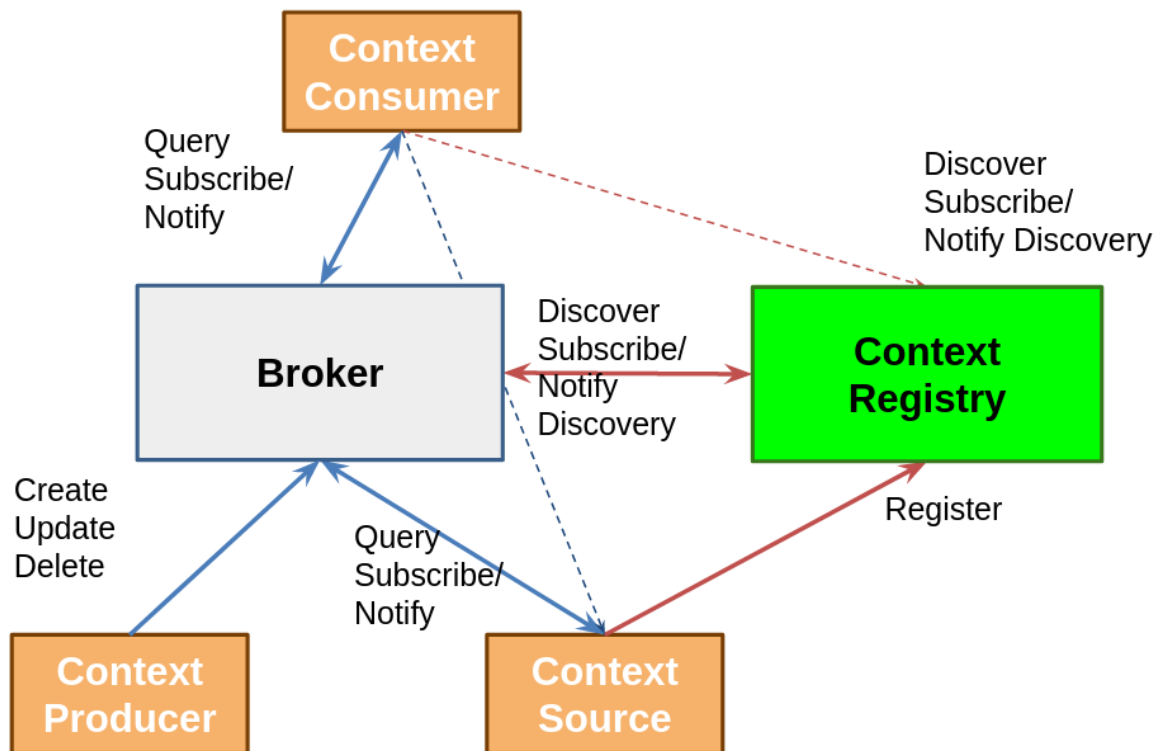


Figure 2: Scenario for broker federation

3.4.4 Multitenancy

In contrast to a federated scenario, multitenancy involves running multiple services within a single instance of a context broker. In this setup, each user group is assigned to a specific tenant. The NGSI-LD implementations (such as Orion-LD, Stellio, Scorpio, etc.) may support multitenancy as an optional feature. To specify a tenant in an HTTP request, the "NGSILD-Tenant" header is used in the HTTP binding.

The creation of tenants can occur implicitly, for example, when a tenant is first used in a create (entity, subscription, registration) operation. There are currently no API operations in NGSI-LD for explicitly creating or deleting tenants. If no tenant is specified, there is always a "default tenant" available. The implementation of tenants, including how isolation is achieved, is up to the specific implementation. Registrations can be targeted to a specific tenant, or if not specified, they will default to the "default tenant". If the same context source/broker needs to be registered for multiple tenants, multiple registrations are required.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	27 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

3.4.5 Distributed operation modes

When multiple brokers are used they can be registered as data sources for others, and several options are available.

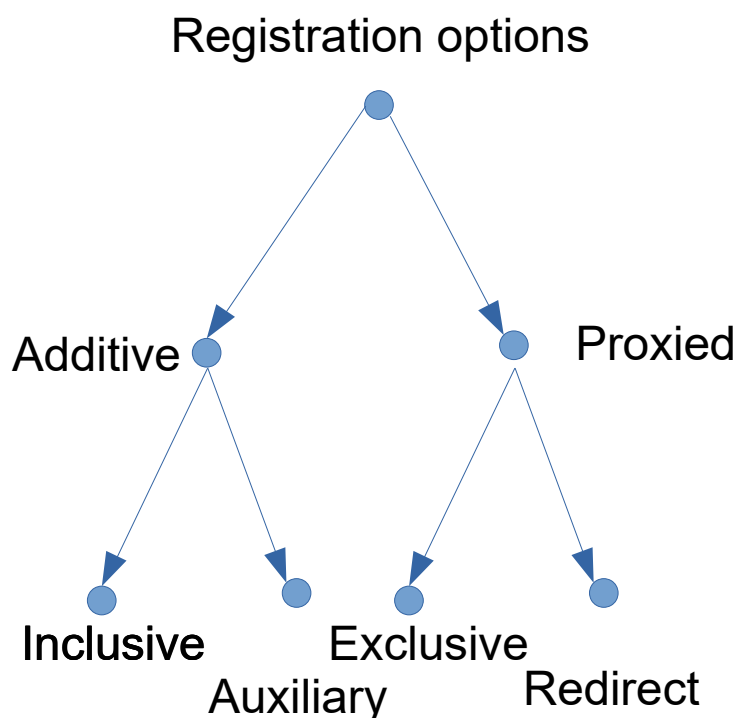


Figure 3: Diagram for distributed operation modes

Additive Registrations

A Context Broker is permitted to hold context data about the Entities and Attributes locally itself, and also obtain data from (possibly multiple) external sources.

Inclusive

An inclusive Context Source Registration specifies that the Context Broker considers all registered Context Sources as equals and will distribute operations to those Context Sources even if relevant context data is available directly within the Context Broker itself (in which case, all results will be integrated in the final response). This is the default mode of operation.

Auxiliary

An auxiliary Context Source Registration never overrides data held directly within a Context Broker. Auxiliary distributed operations are limited to context information consumption operations. Context data from auxiliary context sources is only included if it is supplementary to the context data otherwise available to the Context Broker.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	28 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status: Final

Proxied Registrations

A Context Broker is not permitted to hold context data about the Entities and Attributes locally itself. All context data is obtained from the external registered sources.

Exclusive

An exclusive Context Source Registration specifies that all of the registered context data is held in a single location external to the Context Broker. The Context Broker itself holds no data locally about the registered Attributes and no overlapping proxied Context Source Registrations shall be supported for the same combination of registered Attributes on the Entity. An exclusive registration must be fully specified. It always relates to specific Attributes found on a single Entity.

Redirect

A redirect Context Source Registration also specifies that the registered context data is held in a location external to the Context Broker, but potentially multiple distinct redirect registrations can apply at the same time.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	29 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

4 Operation of the platform

4.1 Operational aspects

From the user requirements emerged some concerns related to the operational aspects of the platform in general, and to cybersecurity more specifically:

- How to trust Open-Source software that is used and integrated into the platform?
- How to deliver an operational, scalable and reactive platform?
- How to ensure the platform stays safe and secure?
- How to monitor the correct behavior of the platform?

This section is organized as follows: Subsection 4.2. describes the quality and security processes to apply during the development, integration and deployment of components inside the FIWARE platform. Subsection 4.3 describes the requirements for a deployment infrastructure that can handle current and future needs of users. Subsection 4.4 describes the security measures to apply to a production environment in operation. Subsection 4.5 describes the security measures to apply specifically to the communication with the legacy systems used by the pilot sites. Subsection 4.6 describes the operation support tools to deploy in order to ensure a correct monitoring of the platform.

4.2 Secure code, from design to delivery

The first concern relates to the trust and confidence that a user may have in a large platform composed from the development and integration of many Open-Source software and libraries.

This is a legitimate concern, and the platform has to define and deploy all the necessary processes and tools in order to ensure the maximum level of security in the software delivery chain.

Thus, we are proposing here a set of security practices to be applied from the design of a new piece of software to its delivery in production.

A new term, Continuous Hacking, started to emerge recently to design this whole process of ensuring the security chain in software development and delivery. It is associated with the STRIDE acronym: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Escalation. The techniques, processes and tools described below follow and address these security topics.

4.3 Secure by design

The first step in this process is to apply the “Secure by design” principles to all the software that is specifically developed in the scope of the GreenMov project. Even considering that most of the software is already available as part of the FIWARE catalog. But the specific requirements of each pilot will comprise some development.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	30 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

It means that the security is considered from the design phase of the application and checked continuously via unit tests focused on security. For instance, if the application receives some user input, it implies to sanitize the data and remove any potential malicious characters.

For this, a minimal and recommended practice is to follow the Open Web Application Security Project (OWASP) top 10 most critical web applications security risks that directly apply to the phase of code design.

To help in these tasks, it would be assessed if it would be necessary to proceed with a static analysis security testing (SAST). A very valuable starting point is the community list of such existing tools that is maintained by the OWASP. What's more, it is expected that the selected tool cover at least the following topics:

- Support a rich variety of languages, and at least all the languages used in the components of the platform.
- Detect the security vulnerabilities
- Integrate seamlessly in a CI/CD chain

4.4 Dependencies scanning

Nowadays, a typical application or microservice in production has 80% of its source code coming from integrated third-party libraries (which in turn have their own dependencies and so on and so forth). This general principle also applies to the components of the FIWARE catalog.

It is thus very important to integrate a dependency scanning process in order to detect as soon as possible a security vulnerability introduced by one of these third-party libraries. What's more, to be effective, it has to be integrated into the whole software development life cycle: new source code added, deployment pipeline, external contributions received via a pull request, ...

As of now, some tools have been identified for a careful evaluation (but larger research will be conducted):

- Dependabot, a service provided by GitHub.
- Integrated security alerts in GitHub projects, as recently made available by GitHub.
- Snyk.

To be valuable, the security scanning of dependencies has to be part of an automated and continuous process, with automatic fixes (or suggestions for fixes at least, via pull request for instance) as much as possible. Thus, it has to be run automatically on a regular basis (for instance, on each pull request, on each commit on the main branches, ...) and to be followed by immediate actions when this is possible (for instance, a deployment of the platform in production if a critical vulnerability has just been fixed).

4.5 DevSecOps

DevSecOps is an extension of the now classical DevOps paradigm. This term is used to emphasize that security must be a core part of the software delivery chain and thus must be deeply integrated into the continuous integration and continuous deployment pipelines.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	31 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

For the continuous integration pipeline, it implies at least to cover the following topics:

- Run static analysis security testing.
- Run security focused unit tests.
- Scan for the security of dependencies.
- Secure the Docker containers.

The first three points being covered above, the following will specifically address the Docker containers security. GreenMov developers will assess if such devsecops would be required for the current integration.

Docker security

The Docker containers security is a large topic by itself. Docker technology is something relatively new, but very largely widespread. Unfortunately, the security aspect of the containers has not been really addressed from the beginning and there is now a large surface left for attacks. A lot has been done in the past months and there are now mature tools and practices to help in dealing with security in a containerized world. This security field is improving and extending every day, as emphasized for instance by the recent announcement of a partnership between Snyk and Docker to improve the overall security of Docker containers and integrate this concern at the heart of a software delivery chain.

The Docker containers security can be roughly divided into:

- Container creation best practices

A lot of practices have emerged recently in this field. They range from best practices at the creation time of a container, to the need to run Docker containers as a non-root user.

These practices will be thoroughly studied and integrated when wiring up the Docker containers composing the GreenMov platform.

Complementary to this, tools that help in checking and enforcing these best practices will be used when it is possible to automate the checking (for instance, a tool like Docker Bench Security (<https://github.com/docker/docker-bench-security>) may be of great value).

Also, new emerging techniques like Buildpacks (<https://buildpacks.io/>) from the Cloud Native Computing Foundation will be considered seriously. Indeed, they provide a higher level of abstraction for building apps compared to Dockerfiles and thus bring a new experience into bridging the gap between the source and the Docker packaging of an application and applying best of breed practices in modern container standards. It also ensures that applications meet security and compliance requirements without developer intervention.

- Container security scanning

As of now, some tools have been identified but larger research will be conducted before a final choice: Clair and MicroScanner.

- Image signing

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	32 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

In order to bring confidence in the images used, Docker provides tools and practices to apply and check image signing, as it is for instance already done in packages distributed on Linux distributions. Such image signing will be applied to each image produced by the platform.

For the continuous deployment pipeline, it implies at least to cover the following topics:

- Dynamic analysis security testing (DAST)

In the same way that a static security analysis is performed on the source code during the continuous integration phase, a dynamic security analysis is potentially an option to be adopted at GreenMov developments during the continuous deployment one.

This time, the analysis is performed on the running platform, typically deployed in a dedicated environment, but with a security configuration that has to be the same as the production environment.

For this specific task, different existing Open-Source tools will be evaluated, and a choice will be made for a proven mature solution. Once again, the OWASP site lists some mature solutions and a tool like Zaproxy, also known as OWASP ZAP, has already been identified as a valid candidate.

- Penetration testing

Another very valuable and critical kind of testing is penetration testing. This is in particular a critical point to be addressed for a smart city management platform that may be subject to cyberattacks, due to the potentially sensitive nature of the underlying infrastructure.

This is a specific field that is well covered and understood. There already exists tools and procedures that will be applied on the platform to be deployed. Security assessment tools, like the aforementioned Zaproxy, can also be used to help and ensure the platform meets the expected security requirements.

- Chaos engineering

Chaos engineering is quite a new field, not directly related to the platform security, but more to the resilience of the platform.

It has gained a lot of popularity some years ago when Netflix released the now famous Chaos Monkey project. Aimed at running against a production platform, it tries to “inject” some abnormal behavior inside the platform (network outage, application failures, ...) in the objective to test the application resilience against a bunch of different external or internal factors. Due to the potential criticality of the software that is going to be deployed on the GreenMov platform, this is an aspect to be considered.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	33 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

5 Data Architecture

This section has the same contents as the equivalent section in deliverable 4.1 because there has not been relevant changes during the period between release of the deliverable 4.1 and 4.2. It could be omitted but it is included here just for making the deliverable 4.2 more self-contained.

5.1 Data Storage architecture and technical format

As long as NGSI-LD is the chosen common standard within GreenMov services and components for interchanging information between the main systems (but for the sensors) the common standard for document sharing and eventually for some of the data storage will be JSON (and specifically JSON-LD).

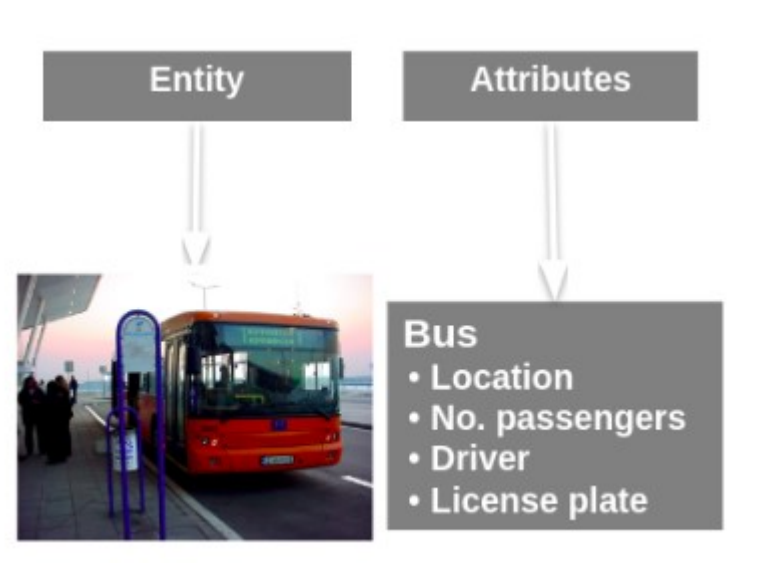


Figure 4: Entity and attributes

```
{
  "id": "ngsi-ld:BUS:001",
  "type": "Bus"
  "location": [215, 33.4],
  "driver": "ngsi-ld:DRIVER:002",
  "licensePlate": "4536KVM"
}
```

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	34 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

The entity¹ could be represented in JSON payload this way.

The information will be split into elements named entities that can be a single JSON payload. the payload is a series of keywords (attributes) and a attached value that can be a single value (e.g. string , data-time, number) or complex values like an array or a more complex object. These entities have a unique attribute, the identifier or ‘id’, that allows to reference them uniquely, and another attribute, the type, which determines their class.

Entities belonging to the same class can have different internal structures because do not need to have proper values for all the attributes These elements can include references to other elements store across the different systems.

5.2 Basic data classes / entities

Although the different pilots would share the same technical architecture there would be quite limitation if they do not share the data structures. Thus, in activity 2 a set of shared data models have been defined so every entity can be shared between the different pilots.

The full list of entities is defined in deliverable 2.2, section chapter 4 Data Models for GreenMov use cases. Here is just the plain list of those data entities created explicitly for GreenMov project.

- [AirQualityForecast](#). A forecast of air quality conditions valid during a period
- [BicycleParkingStation](#). Bicycle Parking Station Schema meeting Passenger Transport Hubs AP Schema specification
- [BicycleParkingStationForecast](#). Bicycle Parking Station Schema meeting Passenger Transport Hubs AP Schema specification
- [NoiseLevelObserved](#). An observation of those acoustic parameters that estimate noise pressure levels at a certain place and time.
- [NoisePollutionForecast](#). Noise Pollution forecast stores the expectation about noise pollution based on some input elements and the noise elements present.
- [ResourceReport](#). Resource Report Schema meeting Passenger Transport Hubs AP Schema specification. A summary of resources connected to mobility services based on defined filters by the person requesting the report.
- [ResourceReportForecast](#). Resource Report Forecast Schema meeting Passenger Transport Hubs AP Schema specification. A summary of the expectations of the resources connected to mobility services based on defined filters by the person requesting the report.

¹ This entity is described here only for explanatory purposes, and it does not correspond with any of the real ones in the use cases. worth to be noted that the value for the attribute driver can be the pointer to a different entity of the type of drive with their own attributes, allowing to create relationships between the entities stored in the context broker

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	35 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- [TrafficEnvironmentImpact](#). Environmental Impact of traffic based on the vehicles traffic and their emission characteristics
- [TrafficEnvironmentImpactForecast](#). Environmental Impact of traffic based on the vehicles traffic expectations and their emission characteristics

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	36 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

6 Conclusions

This document provides general guidance for designing and deploying the technical reference architecture of the GreenMov Project use cases. The reference architecture outlined here will enable the use cases to meet current project needs and adapt to future requirements. A more in-depth analysis of system scalability is included in deliverable 4.3.

The reference architecture offers a comprehensive description of the components, based on the existing reference architecture for smart cities by the FIWARE community, with modifications to fit the GreenMov use cases. The document also includes instructions for installation, with links to more information about each component in different sections.

It's worth mentioning that some components have been added, such as the LDES integration, while others are not currently necessary for the use cases but may be useful in the future. This component could result in a candidate to a generic enabler, as new component, for the FIWARE community.

Further guidance is provided in chapters 3 and 4 on how to deploy and operate the components, with a separate sub-chapter dedicated to security. Some deployment details are located in the annexes to keep this chapter brief. The different options for deploying a network of brokers to retrieve information are also discussed.

The data architecture is only briefly outlined, as the details about the entities containing information are part of Activity 2's scope, and only the new entities created for the project are listed. As mentioned previously, one outcome of the project may result in the creation of a specific generic enabler for sustainable mobility. A generic enabler is an open-source software component validated by the FIWARE Foundation as part of the global framework. Consequently, the requirements for becoming such a component are also included in the annexes.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	37 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

7 References

- [1] <https://github.com/telefonicaid/fiware-orion>
- [2] <https://github.com/FIWARE/context.Orion-LD>
- [3] <https://github.com/ScorpioBroker/ScorpioBroker>
- [4] [1] <https://github.com/stellio-hub/stellio-context-broker>
- [5] [1] <https://www.etsi.org/committee/cim>
- [6] According to Gartner, referenced by <https://www.linkedin.com/pulse/what-context-brokers-alvaro-martin/> A context broker is a service that is designed to gather reachable context data of a variety of types, sources and velocity. It then applies conditioning, integration, rules and analytics to derive the reduced prepared context data, actionable at a point of business decision by a system or a human
- [7] <https://github.com/TREEcg/ngsi-lides>
- [8] <https://github.com/telefonicaid/iotagent-json>
- [9] <https://github.com/telefonicaid/lightweightm2m-iotagent>
- [10] <https://github.com/telefonicaid/iotagent-ul>
- [11] <https://github.com/Atos-Research-and-Innovation/IoTagent-LoRaWAN>
- [12] <https://github.com/telefonicaid/sigfox-iotagent>
- [13] <https://github.com/telefonicaid/iotagent-node-lib>
- [14] <https://github.com/telefonicaid/fiware-cygnus>
- [15]]<https://quantumleap.readthedocs.io/en/latest/>
- [16] <https://github.com/ging/fiware-cosmos>
- [17] <https://fiware-idm.readthedocs.io/en/latest/>
- [18] <https://github.com/conwetlab/FIWARE-CKAN-Extensions>
- [19] <https://github.com/FIWARE/context.Orion-LD/blob/develop/doc/manuals-ld/installation-guide.md>
- [20] https://scorpio.readthedocs.io/_/downloads/en/stable/pdf/
- [21] https://fiware-cygnus.readthedocs.io/en/latest/cygnus-ngsi-ld/quick_start_guide.html
- [22] https://www.keycloak.org/docs/16.1/server_installation/
- [23] <https://github.com/ging/fiware-idm/tree/master/extras/docker>
- [24] https://fiware-idm.readthedocs.io/en/latest/installation_and_administration_guide/installation/index.html
- [25] <https://fiware-pep-proxy.readthedocs.io/en/latest/>
- [26] <https://authzforce-ce-fiware.readthedocs.io/en/latest/InstallationAndAdministrationGuide.html>

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	38 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- [27] <https://fiware-requirements.readthedocs.io/en/latest/>
- [28] <https://fiware-requirements.readthedocs.io/en/latest/development/index.html#documentation>
- [29] <https://readthedocs.org/>
- [30] <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>
- [31] <https://github.com/ralsina/rst-cheatsheet/blob/master/rst-cheatsheet.rst>
- [32] <https://bestpractices.coreinfrastructure.org/en/signup>
- [33] <https://openssf.org/>
- [34] <https://github.com/OAI/OpenAPI-Specification>
- [35] <https://github.com/FIWARE-Ops/marinera>
- [36] <https://fiwaretourguide.readthedocs.io/en/latest/security/introduction/>

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	39 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Annex I. Requirements for a generic enabler

The official site for the complete list of requirements and the process to achieve it is available at the FIWARE site [27]. Main ones are summarized in the coming sections.

Licensing and open SSF Best practices signature

Every Generic Enabler **MUST** comply with the Licensing and IPR Management requirements. Summarizing.

- The source code of the product **MUST** be licensed under one of the well-recognized open source licenses approved by the Open Source Initiative.
- The open-source license under which source code of the product is licensed **MUST** be clearly mentioned in a first-level section of the README.md file included in the main GitHub repository.
- When using a copyleft open-source license, a specific explanatory paragraph of legal opinion **MUST** be added in the section where the open-source license is mentioned
- The legal opinion paragraph above **SHOULD** be accompanying the text describing the adopted open-source license in the headers of all source code files for the product.
- Every enabler **MUST** be open to third party contributions. All offered contributions **MUST** be reviewed within a "reasonable" time frame.
- There **MUST** be a document (CONTRIBUTING.md guidelines) clearly describing the terms under which the IPR of contributions to the source code of the product will be managed. Such document **MUST** be made accessible in (or map to) a first-level section of the README.md file included in the associated GitHub repositories.
- The CONTRIBUTING.md guidelines **MUST** include the template of the Contribution License Agreement for individuals and entities contributing code to the component. As a reference for producing these templates, the following templates derived from the Harmony Agreements project are provided:
 - Individual CLA
 - Entity CLA
- When using a copyleft open-source license, IPR Management rules for contributions **MUST** include clauses as follows:
 - There should be at least one organization which can exercise IPRs on the whole software.
 - There is a commitment to transfer to the FIWARE Foundation the IPRs on the whole software in case that the software is no longer supported by the organization(s) that currently own(s) IPR on the whole software.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	40 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

General requirements

A FIWARE Generic Enabler MUST fit well in the architecture of a “Powered by FIWARE” solution:

- Integrate well with architectures where context management is cornerstone and addressed using FIWARE NGSI (currently FIWARE NGSIv2, compliant with ETSI NGSI-LD in the future).
- Be able to fit within one of the defined FIWARE chapters.

Code control tool requirements and public backlog

GitHub and GitHub Issue tracking MUST be used.

Documentation requirements

A generic enabler to be accepted needs to have accurate, current Documentation MUST be available on Read the Docs and as GitHub content. To guarantee that documentation is of high quality, development related documents MUST be peer-reviewed, and QA verified. See Documentation Guidelines [28] for the best documentation practices. Should you want to benefit from automatic documentation generation systems, namely, Read the Docs [29], you MUST use an approved markup notation:

- Markdown [30] is preferred for simple documents.
- Restructuredtext [31] is an acceptable alternative for complex documentation.

Development requirements

Every Generic Enabler must sign-up to the OpenSSF Best Practices Badge Program [32] and display the badge.

The Open-Source Security Foundation (OpenSSF) [33] Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice. API Specifications MUST be provided. Preferred format is OpenAPI1, a.k.a. Swagger, format.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	41 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final


Annex II. An example of configuration

Further information can be found in the repository of Marinera [35]

How to setup

When following all described steps, the resulting cluster will be a fully working copy of the Kubernetes Clusters, operated by the FIWARE Foundation. If a cluster is already available, you can skip the steps that are already fulfilled. Be aware that this might require changes to the following steps, depending on the degree of deviation from the proposed setup.

1. [Prepare AWS account](#)
2. [Install OpenShift cluster](#)
3. [Install certificates](#)
4. [Install ArgoCD](#)
5. [Prepare ArgoCD for namespaced deployments](#)
6. [Deploy namespaces](#)
7. [Deploy bitnami/sealed-secrets](#)
8. [Create secrets](#)
9. [Deploy MongoDB](#)
10. [Deploy Orion-LD](#)

 For a better understanding of the process, all application-deployments (starting at step 6.) are executed through the ArgoCD-UI. However, all of them also can be done through the argocd-cli. See the cli-installation documentation for that.

1. Prepare AWS account

In order to use the OpenShift installers, provided by RedHat, an AWS account is required: <https://aws.amazon.com> The account needs to be prepared, following the steps described in OpenShift - configure AWS account.

2. Install OpenShift cluster

The process of creating an OpenShift cluster at AWS is described in the OpenShift documentation: https://docs.openshift.com/container-platform/4.7/installing/installing_aws/installing-aws-default.html#installing-aws-default Choose the right method for the used operating system and carefully follow the instructions.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	42 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

3. Install certificates

In order to have proper certificates available for the cluster, we are using Let's encrypt to generate our cluster certificates.

! The following method requires an existing connection to the OpenShift cluster. If you followed the previous steps, this should already exist. If not, install the OpenShift-client and login to the cluster as described in the OpenShift-CLI documentation. Check the connection via `oc whoami --show-server`, the url should match with the cluster you want to use.

! The following steps describe certificate generation for AWS installations. For other cloud-providers, check the options in the acme.sh repo&documentation

Clone the acme.sh github-repo

```
cd$HOME git clone https://github.com/acmesh-official/acme.sh
```

Setup AWS credentials

The acme.sh client requires access to AWS Route53. Create and (locally) store your credentials following the documentation. Create environment variables to be used by the client via:

```
export AWS_ACCESS_KEY_ID=<KEY_ID_OBTAINED_FROM_AWS>
export AWS_SECRET_ACCESS_KEY=<SECRET_OBTAINED_FROM_AWS>
```

Obtain certificates

```
# export information from the cluster, to be used by the acme-client
export LE_API=$(oc whoami --show-server | cut -f 2 -d ':' | cut -f 3 -d '/' | sed 's/-api././')
export LE_WILDCARD=$(oc get ingresscontroller default -n openshift-ingress-operator -o jsonpath='{.status.domain}')
# run acme-client for aws
aws$HOME/acme.sh/acme.sh --issue -d ${LE_API} -d *.${LE_WILDCARD} --dns dns_aws
#export CERTDIR=$HOME/certificates mkdir -p ${CERTDIR}$HOME/acme.sh/acme.sh --install-cert -d ${LE_API} -d *.${LE_WILDCARD} --cert-file ${CERTDIR}/cert.pem --key-file ${CERTDIR}/key.pem --fullchain-file ${CERTDIR}/fullchain.pem --ca-file ${CERTDIR}/ca.cer
```

Create the secrets

```
# create secret for default ingress-controller
oc create secret tls router-certs --cert=${CERTDIR}/fullchain.pem --key=${CERTDIR}/key.pem -n openshift-ingress
# create secret for the api-server
oc create secret tls api-certs --cert=${CERTDIR}/fullchain.pem --key=${CERTDIR}/key.pem -n openshift-config
```

Patch ingress-controller and api-server

```
# patch ingress controller
oc patch ingresscontroller default -n openshift-ingress-operator --type=merge --patch='{"spec": { "defaultCertificate": { "name": "router-certs" } } }'
# patch api-server
oc patch apiserver cluster --type merge --patch='{"spec": { "servingCerts": { "namedCertificates": [ { "names": [ "${LE_API}" ], "servingCertificate": { "name": "api-certs" } } ] } }' }
```

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	43 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Update kubeconfig

The kubeconfig, generated in the installation process of OpenShift, contains a CA for the self-signed certificate generated by OpenShift. This will result in certificate errors when connecting the cluster. Since the Let's encrypt certificate is already trusted, we can remove the CA and use the already installed certs. The kubeconfig is located in the folder created during the cluster setup a look similar to:

```
vi <INSTALLATION_FOLDER>/auth/kubeconfi apiVersion: v1 clusters: - cluster: certificate-authority-data:
<BASE_64_ENCODED_CA> server: https://api.firmware-dev-aws.firmware.dev:6443 name: api-firmware-dev-aws-
firmware-dev:6443 .....
```

The certificate-authority-data entry can simply be removed.

Verify success

```
# check api-server certificate curl -X GET --silent -vvI $(oc whoami --show-server)2>&1| grep issuer # check
ingress-controller curl -X GET --silent -vvI https://$(oc get routes console -n openshift-console -o json | jq -r
'.spec.host')2>&1| grep issuer # both requests should result in something like:* issuer: C=US; O=Let's
Encrypt; CN=R3
```

A more detailed explanation of the process can be found at the RedHat Blog. Be aware that the described process does not automatically renew the certificates (yet).

4. Install ArgoCD

The only component that needs to be directly installed to the cluster is ArgoCD.

Create namespace

In order to separate concerns inside the cluster, we create a namespace/project for ArgoCD to live in:

```
# namespace creation via kubectl, alternatively `oc new-project argocd` would have the same effect kubectl
create namespace argocd
```

Install ArgoCD operator

In order to install ArgoCD, OpenShift comes with a Community Operator for ArgoCD. To install it, go to the OpenShift console([https://\\$\(oc get routes console -n openshift-console -o json | jq -r '.spec.host'\)](https://$(oc get routes console -n openshift-console -o json | jq -r '.spec.host'))) and navigate to the OperatorHub:

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	44 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

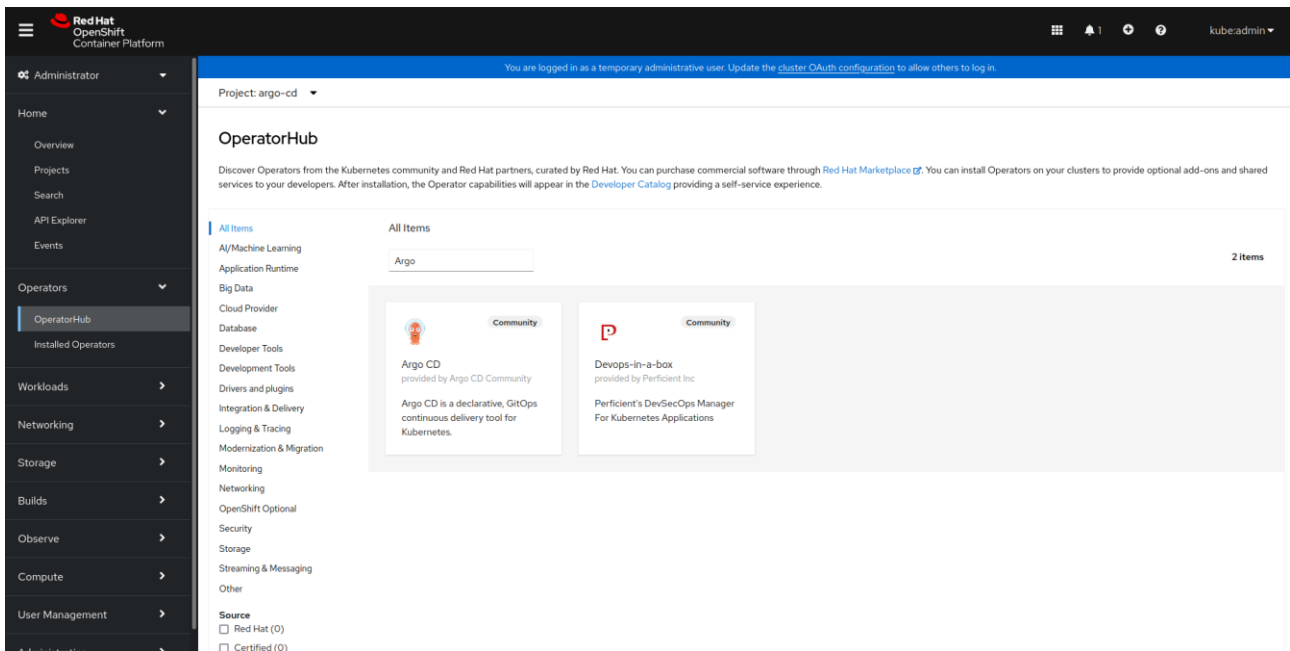


Figure 5: Argo console installation

Search for ArgoCD and follow the installation instructions. Use the "A specific namespace of the cluster" option and choose the namespace created in the previous step("argocd").

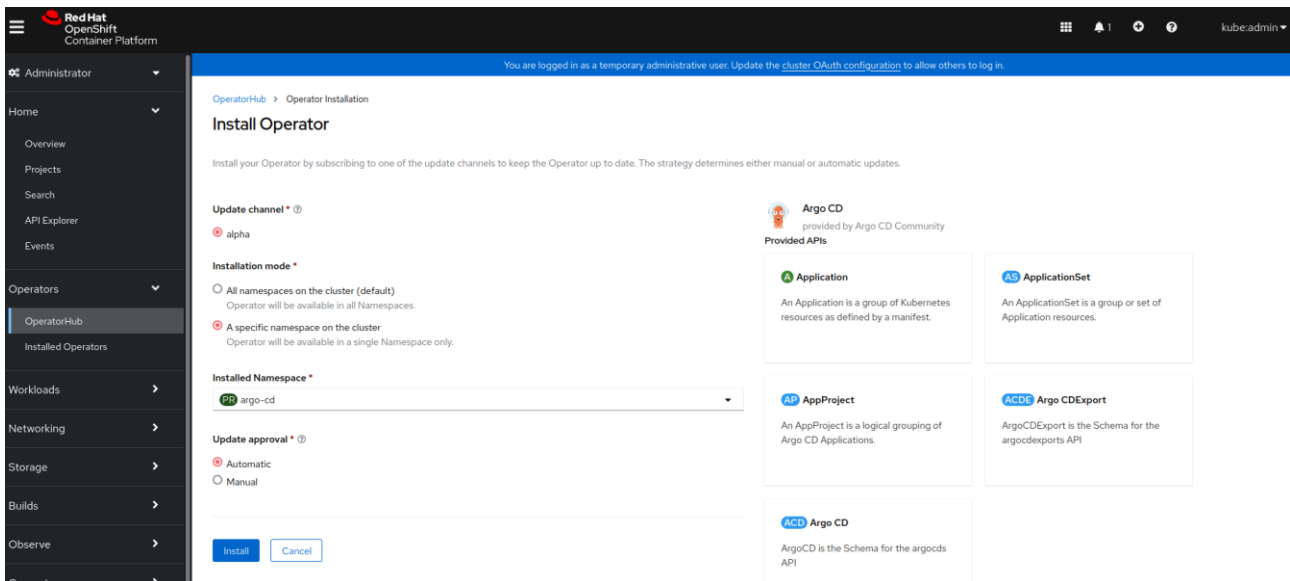


Figure 9: Argo console configuration

Wait for the operator to be installed.

Deploy an instance of ArgoCD

To have a working instance of ArgoCD, we need to instruct the Operator to install one. A definition of our ArgoCD object can be found in the repo under argocd.yaml. Deploy it via:

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	45 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

```
kubectl apply -f argocd.yaml -n argocd
```

After a couple of seconds(probably less than 60), ArgoCD should be available at `kubectl get routes -n argocd -o json | jq -r '.items[0].spec.host'`

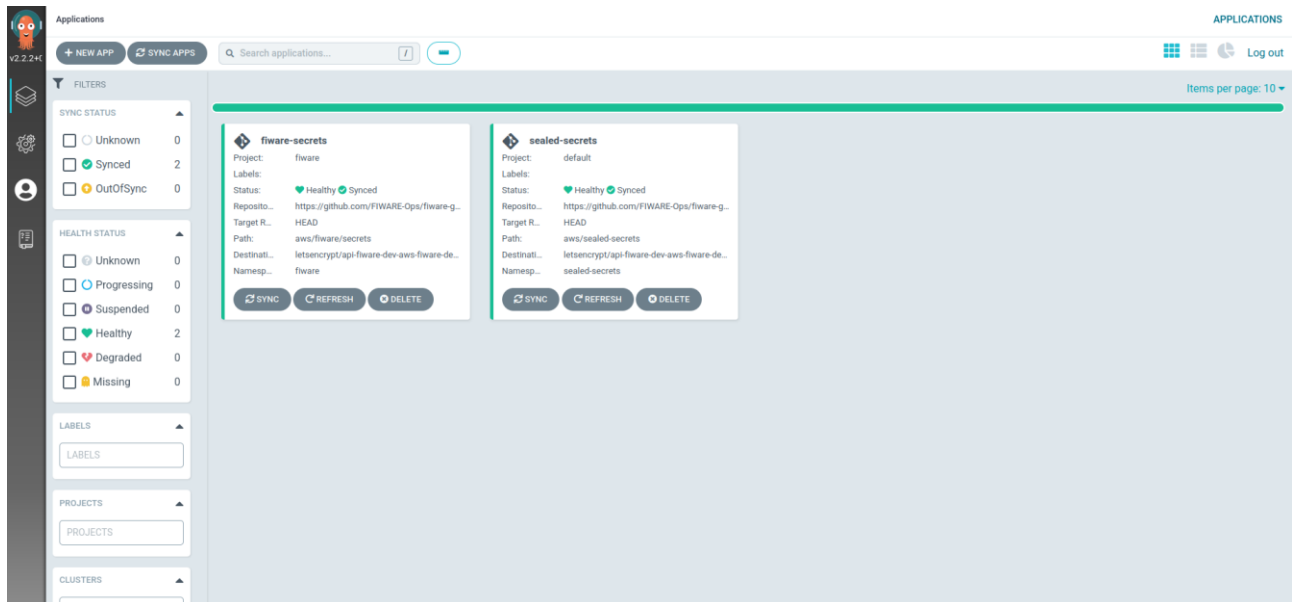


Figure 13: Argo deployment screen

5. Prepare ArgoCD for namespaced deployments

Due to permission restrictions, we need to setup ArgoCD with enough permissions to handle cluster wide deployments.

1. Install the ArgoCD-Client
2. Login with the client:

```
argocd login --sso $(kubectl get routes -n argocd -o json | jq -r '.items[0].spec.host')
```

3. Show available clusters

```
argocd cluster add
```

4. Add the cluster 'letsencrypt/<CLUSTER_ADDRESS>/system:admin

```
argocd cluster add letsencrypt/<CLUSTER_ADDRESS>/system:admin
```

5. Verify the cluster was added

```
argocd cluster list # result should look similar to SERVER NAME VERSION STATUS https://api.fiware-dev-aws.fiware.dev:6443 letsencrypt/api-fiware-dev-aws-fiware-dev:6443/system:admin 1.22 Successful https://kubernetes.default.svc (1 namespaces) in-cluster Unknown
```

Alternative:

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	46 of 61
Reference:	D4.2	Dissemination:	PU	Version:	1.0
				Status:	Final

If you want to use the in-cluster api via 'https://kubernetes.default.svc', the operator-subscription can be configured to allow namespace-wide permissions:

```
kubectl edit subscriptions -n argocd
```

! in case you have multiple subscriptions inside the argocd namespace, make sure to edit the correct one.

```
apiVersion: v1
items:
- apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: argocd-operator
namespace: argocd
spec:
  channel: alpha
  config:
    env:
    - name: ARGOCD_CLUSTER_CONFIG_NAMESPACES
      value: argocd
  installPlanApproval: Automatic
name: argocd-operator
source: community-operator
sourceNamespace: openshift-marketplace
startingCSV: argocd-operator.v0.2.0
```

With this configuration, the operator considers the namespace argocdas one of argoCD's cluster-wide installation namespaces and (within a couple of seconds) upgrades the in-cluster-cluster to handle all namespaces:

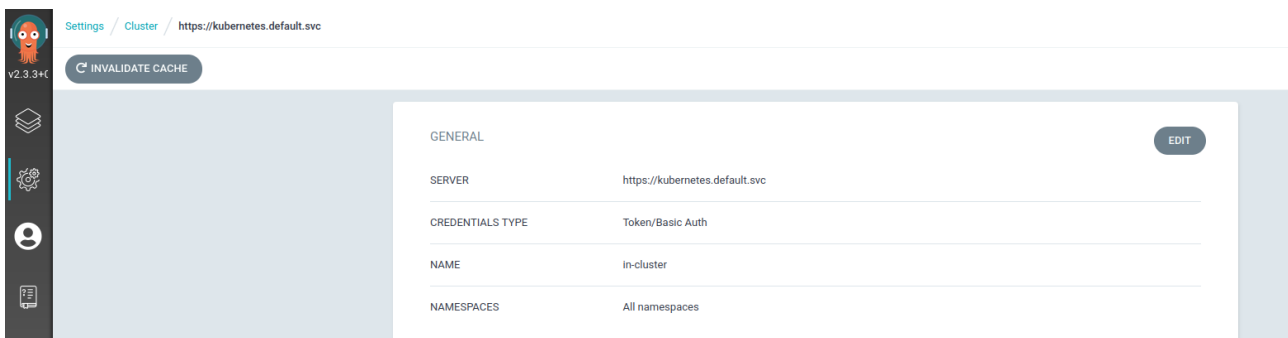


Figure 17: Argo namespaces configuration

6. Deploy namespaces

Since we want to properly separate the workloads in our cluster, we need to manage namespaces. Following git-ops, we will put the namespace-definitions into a repository and let ArgoCD create them for us.

! The following documentation uses the UI to deploy the applications. The same can be achieved via the argocd-cli.

1. Login to ArgoCD

Open `kubectl get routes -n argocd -o json | jq -r '.items[0].spec.host'` in the browser

2. Click on "NEW APP"

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	47 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

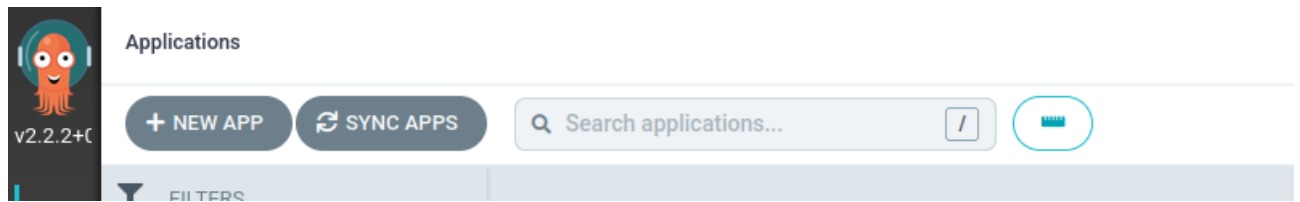


Figure 21: Argo. Creation of new app

3. Fill out the form

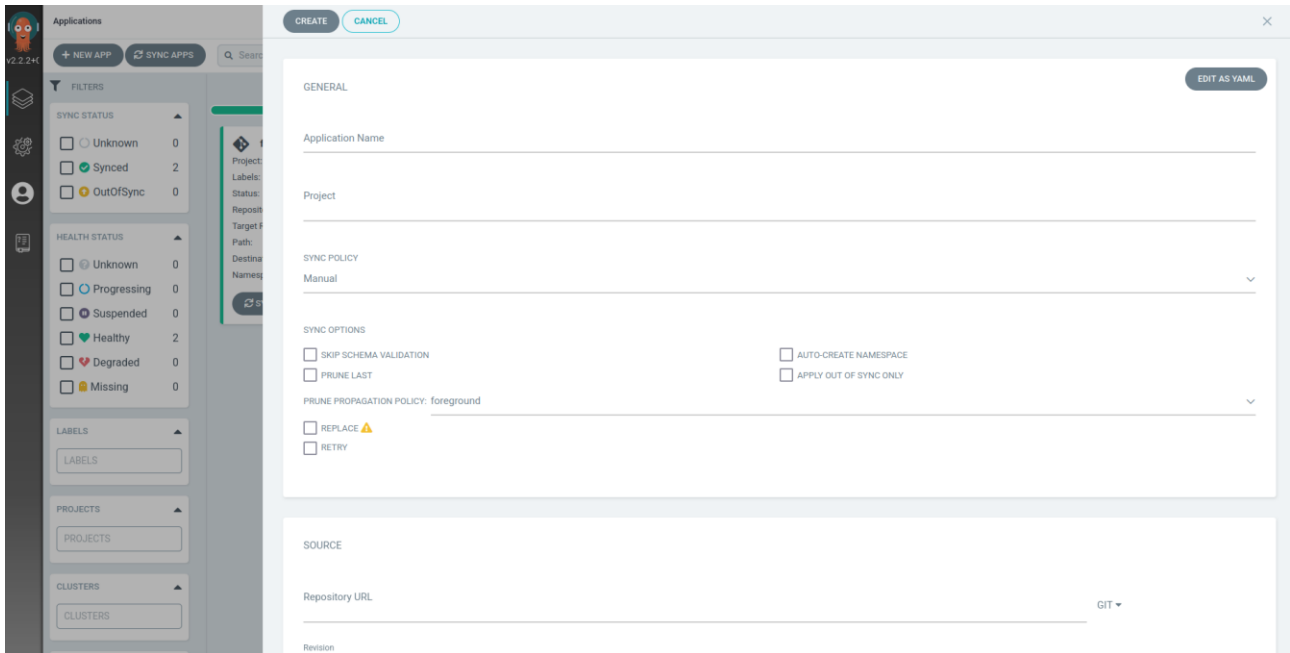


Figure 25: Argo. Configuration of new app

General:

-> Application name: namespaces

-> Project: default

-> Sync Policy: automatic

Source:

-> Repository URL: <https://github.com/FIWARE-Ops/fiware-gitops>

-> Path: aws/sealed-secrets Destination:

-> Cluster URL: -- use the URL of the cluster added via argocd-cli

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	48 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Click create and wait until its running:

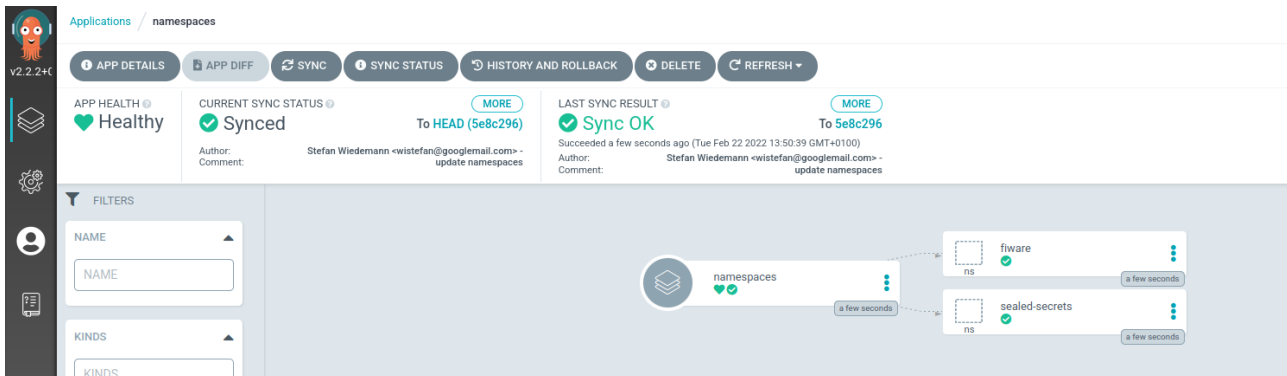


Figure 29: Argo. Monitoring of app

7. Deploy bitnami/sealed-secrets

Using GitOps, means every deployed resource is represented in a git-repository. While this is not a problem for most resources, secrets need to be handled differently. We use the bitnami/sealed-secrets project for that. It uses asymmetric cryptography for creating secrets and only decrypt them inside the cluster. The sealed-secrets controller will be the first application deployed using ArgoCD. Since we want to use the Helm-Charts and keep the values inside our git-repository, we get the problem of ArgoCD only supporting values-files inside the same repository as the chart(as of now, there is an open PR to add that functionality -> PR#8322). In order to workaroud that shortcomming, we are using "wrapper charts". A wrapper-chart does consist of a Chart.yaml with a dependency to the actual chart. Besides that, we have a values.yaml with our specific overrides. See the sealed-secrets folder as an example.

1. Click on "NEW APP"

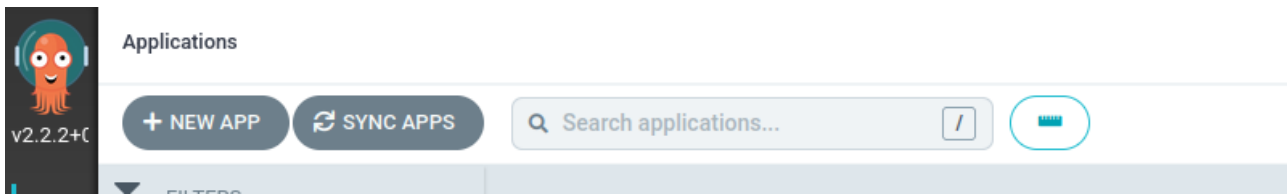


Figure 33: Argo. Create new app

2. Fill out the form

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	49 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

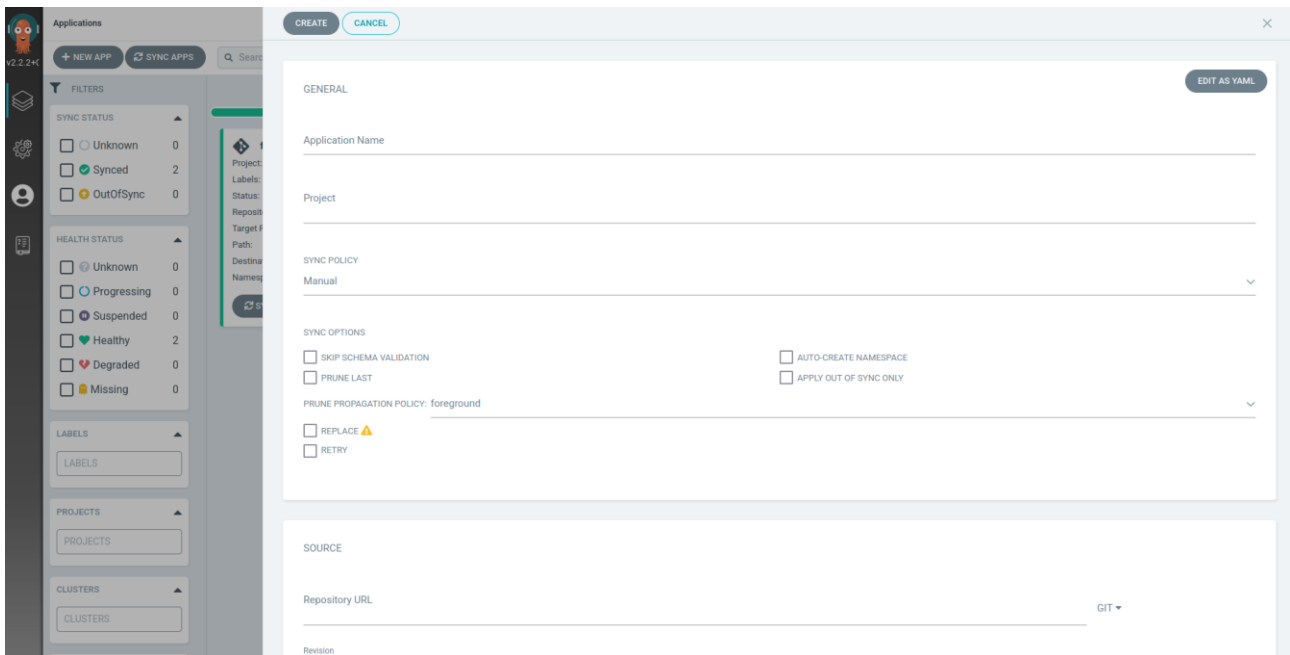


Figure 37: Argo. Creation of new app

General:

- > Application name: sealed-secrets
- > Project: default
- > Sync Policy: automatic

Source:

- > Repository URL: <https://github.com/FIWARE-Ops/fiware-gitops>
- > Path: aws/sealed-secrets Destination:
- > Cluster URL: -- use the URL of the cluster added via argocd-cli
- > Namespace: sealed-secrets Helm: You can provide specific overrides, everything else will be taken from the values-file inside the repository(and thus automatically updated together with the repo).

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	50 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Click create and wait until its running:

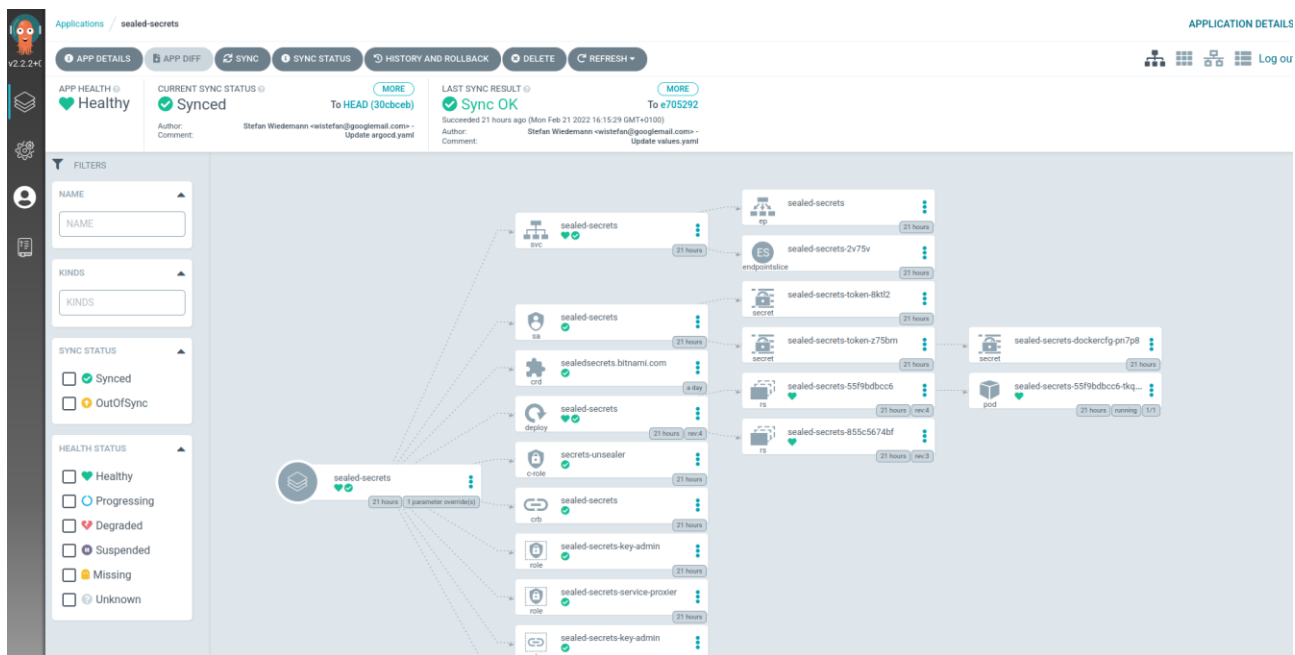


Figure 41: Argo. Monitoring of apps

8. Create secrets

The first applications to be deployed will be the Orion-LD Context Broker together with its MongoDB. In order to communicate in a secure way, the need to use a secret. We will create a secrets-application for our target namespace FIWARE and prepare the secrets via sealed-secrets. For your secrets to be secure, a different repository should be used. The secret-files inside this repository will only work with our cluster, since they can only be decrypted by the sealed-secrets controller they were created at.

1. Create the manifest for the secret(mongodb-secret.yaml) to be used at mongodb(the data-entries need to follow the requirements of the target chart, e.g. bitnami/mongodb):

```
apiVersion: v1
```

```
kind: Secret
```

```
metadata:
```

```
# name of the secret
```

```
name: mongodb-secret
```

```
# namespace the secret should be deployed to - important, sealed-secrets will check the namespace before decryption
```

```
namespace: fiware
```

```
data:
```

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	51 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

```
# the actual data, needs to be base64 encoded
mongodb-password: cGFzc3dvcmQ=
mongodb-replica-set-key: cGFzc3dvcmQ=
mongodb-root-password: cGFzc3dvcmQ=deploy orion-ld
```

⚠ Do never push this file to git. If it happens by accident, try not to remove it but replace all of them.

Install kubeseal

kubeseal is the client-side application for creating the sealed-secrets. Install it, following the official documentation

Seal the secret

The secrets now needs to be encrypted before put into git:

```
# pipe the manifest into kubeseal. We need to specify the controller and its namespace, since we installed it
out of its default location kubeseal <mongodb-secret.yaml >mongodb-sealed-secret.yaml -o yaml --controller-
namespace sealed-secrets --controller-name sealed-secrets
```

The resulting "mongodb-sealed-secret.yaml" will look similar to:

```
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
name: mongodb-secret
namespace: fiware
spec:
encryptedData:
mongodb-password:
AgALc3Y7I6MhLszeRVbfyWnQVi0Jdjrozxyw1syAWRbIzAKsw8TkI1h+6zcUIp7v5U+/G3LZerTZoZyr61c
LXeoBNCXTPH5JDM9lhFcvP2rOfyicEo7E8pAzfsqh9BflUcGhUJADajCtQVhvTonArt+xYsEx0TFs97/Q9
Vp1boj/GyO/vc/9Ly++hs29/Dh1W1QSyNXRs5glZKdGveVJCzQ4iZFF+V6aJfrXUpHNgZyNuMGzpPJlzy6
TpiqnKqu1RoiFCByVazeU5IRi13VAut4W/aFeCEWoaJZhHrWHLxaJWbSKzmR2Lpk48n7e4tBPjFvQPf3Ej
05qdrwTTwKo+TWkSU4DpY307NDO+k0DSOpq3SvZfEQYh3DPAj4grXfyHBXjz9mDmg3ZApztBdxwC
RRIG2Uh3DfY15AkYMWPkqkhisApPJdb8AWjydsEutxf7gc8MLRyYBRrKP7ewJjzGXOs3AGJMzoV3BA/
kK8madk9nyLQGIA0cff4MgTXDe1XCiBUeE/AOlbfE9Z/X/NDUc6P3HGhf6mpvL2V4RxBEMqAc9EVE
mM+LVT40mKXyi7g9oMDDAbY7Mp9XMhY+B2o+IxqeW08kMzyIODMuJ8h9om8A4MW1MrxWpi2P7
SoV4/fRmgetkb1rpabR2Jd0arB+RHEA2/zhwDeDbNGRoNkh9esN1A566ALJO+YxxCyFc3lpNe5eeTqgnDz
59uCFJwauKkc4AzIwbHBmAYnnGfPBrhMOJRNdxJ
```

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	52 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

mongodb-replica-set-key:

```
AgAbieAbFOoVy4IBiNQDU+rmJAre8p3ThmDMzSmnaBOhnKqwYk3zOxcZfEyPKXHfI1PPv8It/H44gzyk
M4NP8Hi9QKjdBqYkvE77zfOLPQAjcbwfxjdpfsUx85n7KcCwFuQLY1Xu/b8G6zxI/+XW/f/sKuiX2qAc
KlzMLk2dAWkeU8TgA+S5jYPPgHyDAEnJaKlRoKq7ADUeQae8Bt9HvQhhNDOEZyhuyLrOo/fTajYXK
2u0maLARf79ja908oBqpc/H5gQAP/Sd3+Q7U6pS+Eo7k8+t1LnG7G+Hbc62aJMoEZO4pJhMyqy+wFjqmZi
hjVJCEkf6qP0TuLBO5tN2EYs5Jr8pegbrCFuUexqf62YItQ+U//24iEUVNrUM9QaBBupCWt0gdoDQTEK5e
6+dyYvf6zmKZw6sfQKYbLNEwzJdJp26K71IBQwGdjmbZIkpBHV7u7QeLO8SG3VaoHOFFHC3vMRE4U
Ad7afwrHuK26Tsd1dU1m1tK9nnwLqR0AoYuHHK7ZQAt1iLog5xuiENIp3K2ZVxzmK+I5J3coE3ic5KST
Ri12fSEaV5Rk504GJQ7O+m75UImdYBe+tvmbvsyAzwkMiJwGxWI9MaKwA8ceKQPldqoilTFTcogQ4dq8
Vw9Zy8JvSmd0NpOZP4xXNQn5K0YpEOBgDN7+U7dm3ar9lO8ErjIhCAszWCHa1ymGInF5tqpKiT30/g
XANjUeTO2fdKs3c/DuEj27Z4M/hVmdhc24pB
```

mongodb-root-password:

```
AgAeMwbU6j+vIUXEgLRVfTscIjC7xHXz8w16qjJzhUbpe7EcXAq24qCzHo0hJA2b3oZmbu6OeNnQALjP
VfUZ5Kz4CiQf8klCe3RECqYbMo+rAslZPclMgnOzuVifNVrbr6W0pvQNnnzr3s+DSINKi9Qudq18qjSrK3h
vjroibB9TF771I5PleptAzdm10Y+kwRTKDTwTqWSFPPzQqJKFE/JAnL/oC+Li4woaDGJTuvEqsf12qrpmEF
+76iCrk2OjMGMVuV+ighIcemI5yUcCvL9DnxbcybA8x2vd6r7p+3ZbQat+6l2FLhTmh/78vwNWKuQyWL
D/gPOqo8VI7tBwX8AQhXfKCHLeBE4DnepGH+r6dzKfBZGXXKokynWXvfcXr4rBI5scfVHIJpagrYOShv
1UisFdV1nD8CVlsQEXZ9ZOdcbm4ZduT3X0OJ7+voocPPEucV1S3HdifPDjA032zHJRAYMBqA+CL2R
Rr+JpnmHVMoPS4f6KT8y3ydadllg7dFSIzyNciSY9uaLDQ28im6fa5aqXoQtKQZmUWSXIa4bd06dHJdygm
+eQUDzxrZmAY48sRi6IvTtTZeU/MKiQDzmTtkBDa1Cimsly2ceMBnFE5FLd/D/aTE2LWTRMrRXCxNk
FiTKf3wE+919HdjgHREAZjtNQd+plQl6fhlaXtIDUtXL5qOkporXACKnykdZdhNYkkMdkBcqjNLpvTZ4
9nsXiTrX8fWT6v29920jTDHuySjSpBZ
```

template:

metadata:

name: mongodb-secret

namespace: fiware

[Push the mongodb-sealed-secret.yaml file to your repository](#)

Click on "NEW APP" - this step will continuously repeat 😊

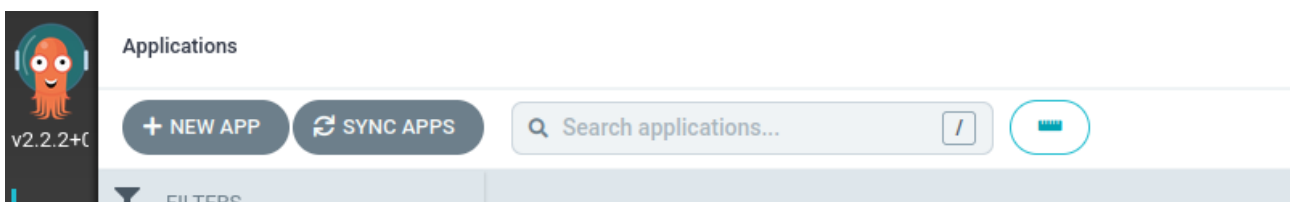


Figure 45: Argo. Creation of new app

Fill out the form - in contrast to "sealed-secrets" this will consist of plain manifests(like "namespaces")

the example will use this repository, please replace with your own

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	53 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

General:

-> Application name: fiware-secrets

-> Project: default

-> Sync Policy: automatic

Source:

-> Repository URL: <YOUR_REPOSITORY>

-> Path: aws/fiware/secrets Destination:

-> Cluster URL: -- use the URL of the cluster added via argocd-cli

NEEDS to be the same as defined in the secret

-> Namespace: fiware

Click create and wait until the sealed-secret is deployed and an unsealed secret is created from it:

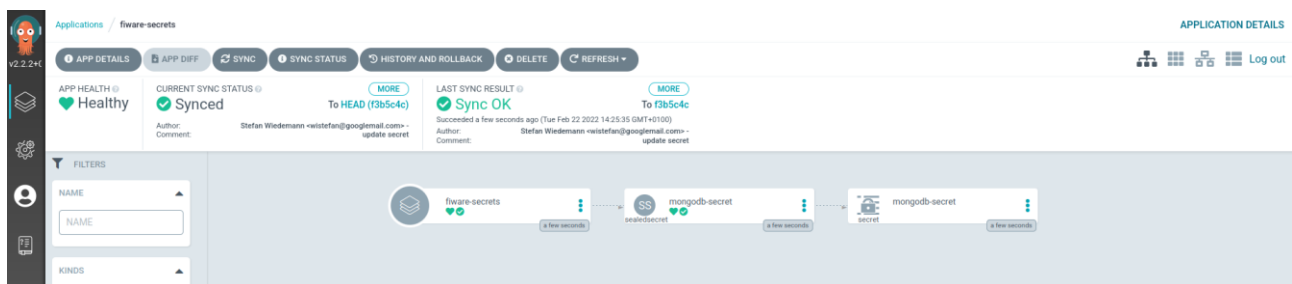


Figure 49: Argo. Secret configured

9. Deploy MongoDB

Deployment of the applications and databases will now all follow the same pattern - create an application in ArgoCD, that references the repository. Check the MongoDB values file to see it referencing the created secret - mongo-db.auth.existingSecret.

Click on "NEW APP"

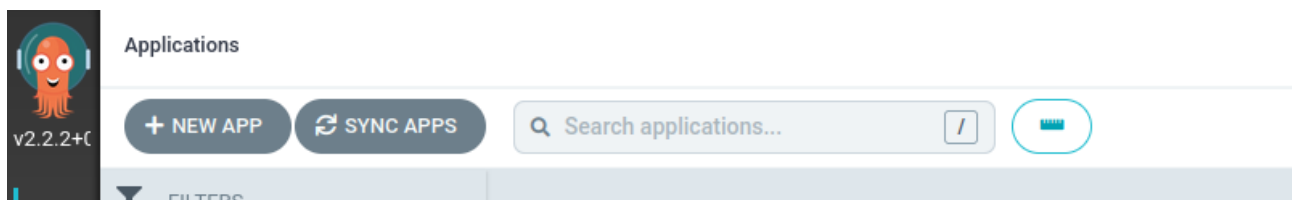


Figure 53: Argo. New app

Fill out the Form

General:

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	54 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

-> Application name: fiware-mongo-db

-> Project: default

-> Sync Policy: automatic

Source:

-> Repository URL: <https://github.com/FIWARE-Ops/fiware-gitops>

-> Path: aws/fiware/mongodb

Destination:

-> Cluster URL: -- use the URL of the cluster added via argocd-cli

-> Namespace: fiware

Helm: You can provide specific overrides, everything else will be taken from the values-file inside the repository (and thus automatically updated together with the repo).

Click create and wait :

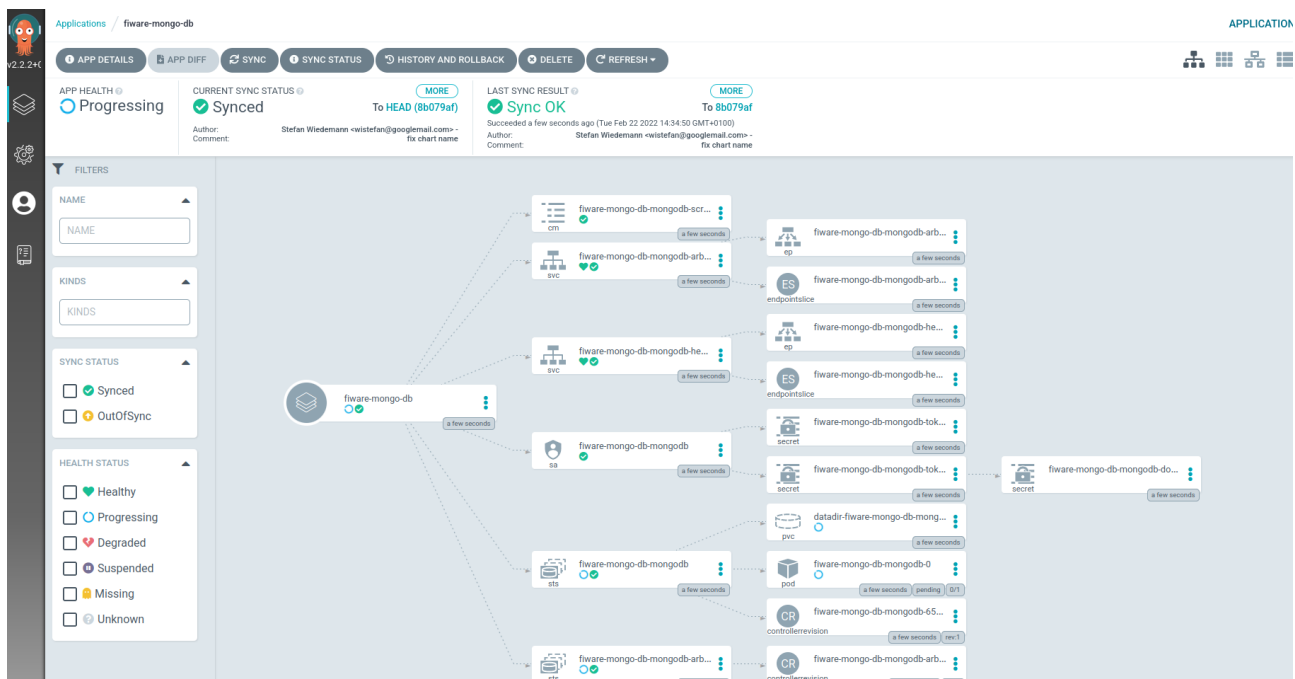


Figure 57: Argo. Deployment of MongoDB

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	55 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

10. Deploy Orion-LD

Click on "NEW APP"

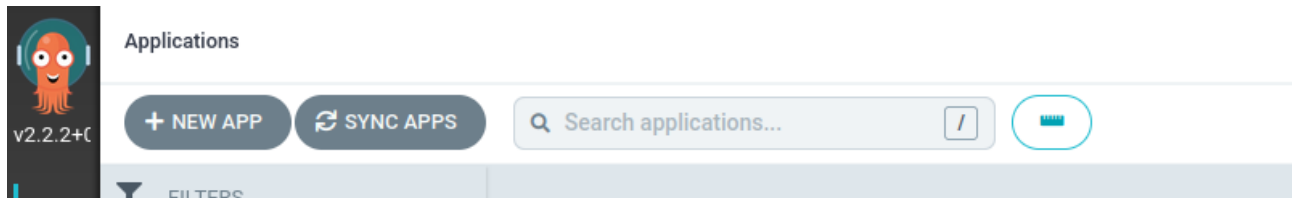


Figure 61: Argo. Creation of new app

Fill out the Form

General:

-> Application name: fiware-orion-ld

-> Project: default

-> Sync Policy: automatic

Source:

-> Repository URL: <https://github.com/FIWARE-Ops/fiware-gitops>

-> Path: aws/fiware/orion-ld Destination:

-> Cluster URL: -- use the URL of the cluster added via argocd-cli

-> Namespace: fiware

Helm: You can provide specific overrides, everything else will be taken from the values-file inside the repository (and thus automatically updated together with the repo).

Advanced topics

In order to further customize deployments more tooling can be added to the cluster:

- [Automatic creation of subdomains and ssl-certificates](#)

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	56 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Annex III. Concepts

Reactive manifesto

The Reactive Manifesto is a set of principles for building reactive systems. A reactive system is a system that is responsive, resilient, elastic, and message-driven. These principles were first outlined in a manifesto that was published in 2013, and they have since been adopted by a growing number of software developers and organizations.

The main principles of the Reactive Manifesto are as follows:

- **Responsive:** A reactive system should respond to user requests in a timely manner, providing feedback and updates to the user as needed.
- **Resilient:** A reactive system should be able to withstand failures and continue to function correctly, even in the face of external factors such as network outages or hardware failures.
- **Elastic:** A reactive system should be able to scale up or down as needed to meet changing demands, without compromising performance or reliability.
- **Message-driven:** A reactive system should use asynchronous message-passing to communicate between components, allowing them to interact in a loosely-coupled and scalable manner.

Overall, the Reactive Manifesto provides a set of guiding principles for building systems that are responsive, resilient, elastic, and message-driven. These principles can help developers to create systems that are better able to handle the challenges of today's complex and distributed environments.

Blue-Green deployment

Blue-green deployment is a technique for deploying software updates. It involves maintaining two identical production environments, called blue and green, and switching traffic between them in order to deploy new versions of the software. This allows you to deploy updates without any downtime, and it makes it easier to roll back to the previous version of the software if there are any issues.

The basic steps for a blue-green deployment are as follows:

- The blue environment is used for production, and the green environment is idle.
- A new version of the software is deployed to the green environment.
- The green environment is tested to make sure that the new version of the software is working as expected.
- Once the green environment has been tested and verified, traffic is switched from the blue environment to the green environment, so that users are now accessing the new version of the software.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	57 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

- The blue environment is then idle, and can be used for the next deployment.

Overall, blue-green deployment is a useful technique for deploying software updates in a way that minimizes downtime and allows for easy rollbacks. It can help to improve the reliability and availability of your software.

Canary deployment

Canary deployment is a technique for rolling out new software versions to a subset of users, before deploying the update to the entire user base. This allows you to test the new version of the software in a production environment, with real users, before making it available to everyone. This can help to ensure that the update is working as expected, and it can give you an opportunity to fix any issues before they affect a large number of users.

The basic steps for a canary deployment are as follows:

- A new version of the software is created and deployed to a small group of users, called the "canary" group.
- The canary group uses the new version of the software, and their usage is monitored to see if there are any issues or problems.
- If the canary group experiences any issues, the new version of the software can be rolled back, and the problem can be fixed.
- If the canary group does not experience any issues, the new version of the software can be deployed to a larger group of users, such as a "beta" group.
- If the beta group also does not experience any issues, the new version of the software can be deployed to the entire user base.

Overall, canary deployment is a useful technique for rolling out new software versions in a controlled and safe manner. It allows you to test new versions of the software with real users, and it provides an opportunity to fix any issues before they affect a large number of users.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	58 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Annex IV. Security

Here there are some generic recommendations regarding security that can be adopted by the use cases in their architectures. Further information can be found in the Security Access and API Management repository [36].

General security considerations

The security of the overall system is a multilayer building, and the precise requirements should be described for each implementation of the reference architecture.

anyhow some elements to build the security of the system has to be in place.

- A proper identification and permission system has to be implemented across the system. This mechanism has to be able to restrict access to non-authorized people to the restricted operations.
- For each component a precise description of the function has to be available.
- Additionally, a feedback mechanism has to be in place, and it has to be proved that it is regularly checked, and the raised issues addressed and eventually solved.
- The process for contribution of new features or fixes has to be clearly explained and the mechanism for contribution also available and clear. Some standards for contribution to be accepted should be also documented (in order to prevent low quality contributions that can drag the overall security of a component). Complementary, a discussion mechanism for new features or approaches to fixes has to be available.
- It is required to have updated documentation of the software
- Last but not least the overall project has to be maintained. (i.e. for basic software updates)
- A unique version number has to identify different version and a proper naming method for versioning should be clearly described and adopted.
- the different software blocks must have an automatic testing mechanism and explanations on how to run it to check proper running.
- Cryptographic protocols and algorithms have to be implemented whenever necessary to ensure security. The default security mechanisms within the software produced by the project **MUST NOT** depend on broken cryptographic algorithms
- Whenever applicable every element of the project **MUST** use a delivery mechanism that counters man-in-the-middle attacks.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	59 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

As popularized by the Kubernetes project, the security of a cloud native platform relies in the security of the 4C's:

- Security of the code
- Security of the container
- Security of the cluster
- Security of the cloud

The security of the code and of the container are already discussed in detail in the previous sections. The security of the cluster is developed in the following paragraphs.

The security of the cloud will have to be evaluated on a case-by-case basis, as the pilot sites emitted the will to host the FIWARE platform on their own premises or within the infrastructure of their usual cloud provider. Security recommendations will be provided and checked all along the deployment to ensure all the platforms are deployed according to the best practices in cloud security, with support from the technical team of the project.

Security of communications

The security of communications applies to different levels:

- First of all, all HTTP communications have to be done through HTTPS (the use of the Certbot certificate provider, which is now well established and largely deployed, will be considered first)
- The communications between the FIWARE platform and the legacy systems will be secured with respect to the security protocols set by each pilot. This is dealt with in the next section.
- The internal communications between the components of the platform should also preferably be secured. This is typically done by using the TLS cryptographic protocol when exchanging data between components, to avoid traffic sniffing
- The communications from and to the sensors, via the IoT Agents. The security of these communications depends on the underlying protocol, so it will be defined and applied on a case-by-case basis.

Management of secrets

Every microservice has to know some passwords, secrets or tokens to communicate with other systems (be it a database, an external service, an authentication provider, and so on). They of course must not be stored in clear text, not even into a private VCS. A first considered step is to use environment variables defined only on target hosts. A more robust approach is to encrypt secrets and use an external service to manage them (for instance HashiCorp Vault or Spring Vault).

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2			Page:	60 of 61		
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final

Slow down attackers

Eventually, an attacker will try to brute force the authentication to the API in order to gain access to the system and expose sensitive or confidential data. One measure to mitigate that risk is to slow down such attacks. This can be done by implementing rate-limiting, whether in the application code or at an API gateway level. It is also more effective if a SIEM tool is deployed inside the platform, for a quicker reaction to such events.

Intrusion detection system

The production also has to be protected from intrusions, that means that an intrusion detection system must be set up (existing tools like Falco, Suricata or else Snort will be considered). This eventually can be completed by a SIEM tool.

Data integrity

Finally, the data at rest in databases has to be encrypted, as it can potentially be leaked in case an attacker gains access to the platform. As this is an expensive process, only sensitive or confidential data will be encrypted. The techniques and algorithms depend on each database vendor thus, it will be checked on a per-database basis, and adapted security measures will be applied on each.

Document name:	D4.2 GreenMov Reference Architecture and guidelines v2				Page:	61 of 61	
Reference:	D4.2	Dissemination:	PU	Version:	1.0	Status:	Final