FIWARE
Global Summit

Leading the digital transformation

OPEN SOURCE | OPEN STANDARDS
OPEN COMMUNITY

Getting started with NGSI-LD

Jason Fox, Senior Technical Evangelist, FIWARE Foundation

Stefan Wiedemann, Technical Lead & Architect, FIWARE Foundation

Gran Canaria, Spain
14-15 September, 2022
#FIWARESummit

FIWARE
Open APIs for Open Minds

Useful links

What is JSON-LD:
https://www.youtube.com/watch?v=vioCbTo3C-4

JSON-LD Core Markup:
https://www.youtube.com/watch?v=UmvWk_TQ30A

Compaction and Expansion:
https://www.youtube.com/watch?v=Tm3fD89dqRE

JSON-LD Playground & examples
https://json-ld.org/playground/

Linked Context Data: NGSI v2 to NGSI-LD

From: https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_faq/index.html

- **NGSI-LD** is an evolution of the FIWARE NGSI v2 information model, and has been updated/improved to support linked data (entity relationships), property graphs and semantics (exploiting the capabilities offered by JSON-LD). This work has been conducted under the ETSI ISG Context Information Management initiative.

```
{
   "@context":  [
      "https://fiware.github.io/data-models/context.jsonld",
      "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
   ],
   "id": "http://dbpedia.org/resource/John_Lennon",
   "type": "Person",
   "name": {"type": "Property", "value": "John Lennon"},
   "born": {"type": "Property", "value": "1940-10-09"},
   "spouse": {"type": "Relationship", "object": "http://dbpedia.org/resource/Cynthia_Lennon" }
}
```

- Creating proper machine-readable Linked Data is **fundamental** to NGSI-LD.
- **NGSI-LD** Payloads are valid **JSON-LD**

What is Core @context?

```
"ngsi-ld": "https://uri.etsi.org/ngsi-ld/",
"geojson": "https://purl.org/geojson/vocab#",
"id": "@id",
"type": "@type",

"Date": "ngsi-ld:Date",
"DateTime": "ngsi-ld:DateTime",
"Feature": "geojson:Feature",
"FeatureCollection": "geojson:FeatureCollection",
"GeometryCollection": "geojson:GeometryCollection",
"LineString": "geojson:LineString",
"MultiLineString": "geojson:MultiLineString",
"MultiPoint": "geojson:MultiPoint",
"MultiPolygon": "geojson:MultiPolygon",
"Point": "geojson:Point",
"Polygon": "geojson:Polygon",

"GeoProperty": "ngsi-ld:GeoProperty",
"Property": "ngsi-ld:Property",
"Relationship": "ngsi-ld:Relationship",

"ContextSourceNotification":"ngsi-ld:ContextSourceNotification",
"ContextSourceRegistration":"ngsi-ld:ContextSourceRegistration",
"Notification": "ngsi-ld:Notification",
"Subscription": "ngsi-ld:Subscription",
```

… etc

```
"coordinates": {
   "@container": "@list",
   "@id": "geojson:coordinates"
 },
"temporalQ": "ngsi-ld:temporalQ",
"throttling": "ngsi-ld:throttling",
"observedAt": {
    "@id": "ngsi-ld:observedAt",
    "@type": "DateTime"
 },
"timeInterval": "ngsi-ld:timeInterval",
"unitCode": "ngsi-ld:unitCode",
"value": "ngsi-ld:hasValue",
"values": {
  "@id": "ngsi-ld:hasValues",
  "@container": "@list"
},
```

… etc

```
"@vocab": "https://uri.etsi.org/ngsi-ld/default-context/"
```

NGSI-LD: Evolution not Revolution

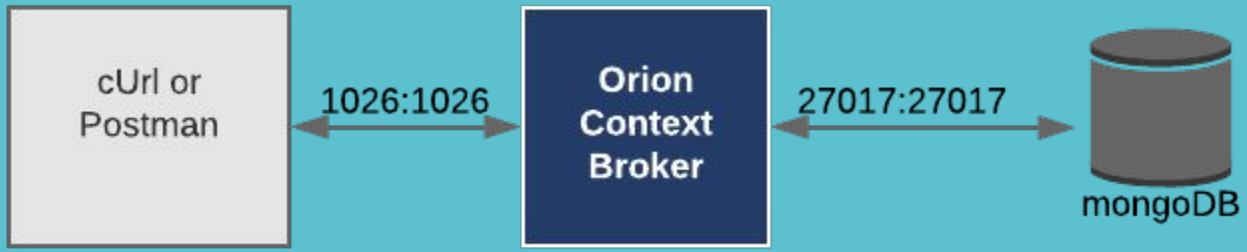
NGSI v2

- Well defined **REST API** for context data using **JSON** payloads.
  GET, POST and other HTTP verbs do the things you expect
- CRUD operations - `/v2/entities` endpoint
- Augment your context data - `/v2/registrations` endpoint
- Push context data to other services - `/v2/subscriptions` endpoint

NGSI-LD

- Well defined **REST API** for context data using **JSON** and **JSON-LD** payloads.
  GET, POST and other HTTP verbs do the things you expect
- CRUD operations - `/ngsi-ld/v1/entities` endpoint
- Augment your context data - `/ngsi-ld/v1/registrations` endpoint
- Push context data to other services - `/ngsi-ld/v1/subscriptions` endpoint

Demo: NGSI-LD - Properties



NGSI-LD Properties: Creating an Entity

NGSI v2

```
curl -iX POST 'http://localhost:1026/v2/entities' \
  -H 'Content-Type: application/json' \
  -d '{
    "type": "Store", "id": "store001",
    "category": { "type": "Array", "value": ["commercial"]},
    "address": { "type": "PostalAddress", "value": {
            "streetAddress": "Bornholmer Straße 65",
           "addressRegion": "Berlin",
           "addressLocality": "Prenzlauer Berg",
           "postalCode": "10439"
        },
        "metadata": {
           "verified": { "type": "Boolean","value": true}
        }
    },
    "location": {"type": "geo:json",
        "value": {"type": "Point",  "coordinates": [13.3986, 52.5547]}
    },
    "name": {"type": "Text", "value": "Bösebrücke Einkauf"}
}'
```

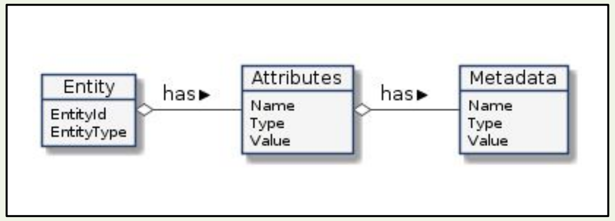
NGSI-LD

```
curl -iX POST http://localhost:1026/ngsi-ld/v1/entities \
  -H 'Content-Type: application/ld+json' \
  -d '{
    "type": "Building", "id": "urn:ngsi-ld:Building:store001",
    "category": { "type": "Property", "value": ["commercial"]},
    "address": {  "type": "Property"," value": {
           "streetAddress": "Bornholmer Straße 65",
           "addressRegion": "Berlin",
           "addressLocality": "Prenzlauer Berg",
           "postalCode": "10439"
        },
        "verified": { "type": "Property", "value": true }
    },
    "location": { "type": "GeoProperty",
        "value": { "type": "Point",  "coordinates": [13.3986, 52.5547]}
    },
    "name": { "type": "Property", "value": "Bösebrücke Einkauf" },
    "@context": [
        "https://fiware.github.io/data-models/context.jsonld",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}'
```

NGSI-LD Properties: Data Model

The NGSI LD data model is more complex; the definitions of use are more rigid which lead to a navigable knowledge graph.

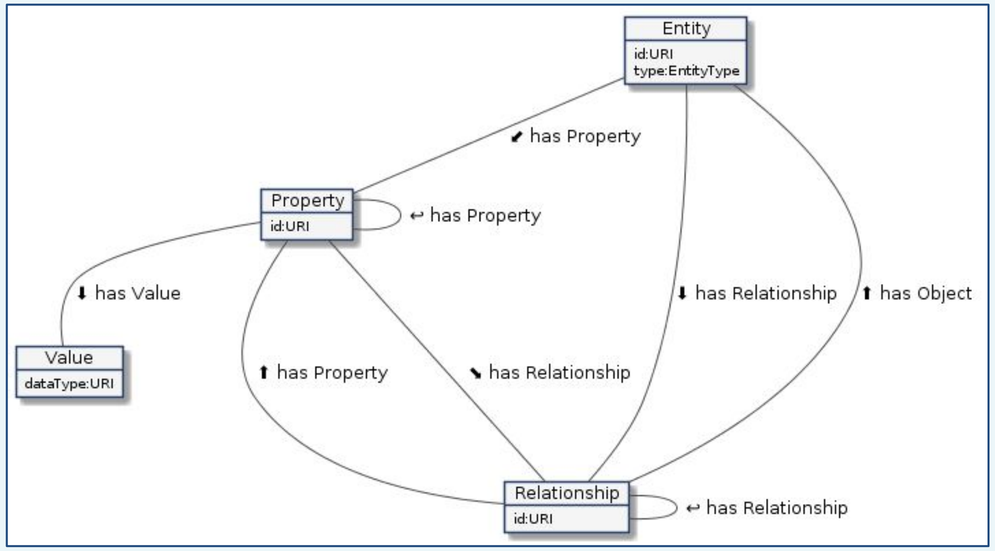
NGSI v2



- Entities
- Attributes
- MetaData

NGSI-LD

- Entities
- Properties
- Relationships
- Values



plus …

- Properties of Properties
- Properties of Relationships
- Relationships of Properties
- Relationships of Relationships

.

plus …

- Properties of Properties of Properties
- Relationships of Properties of Properties
- Properties of Properties of Relationships
- Relationships of Properties of Relationships
- Properties of Relationships of Properties
- Relationships of Relationships of Properties
- Properties of Relationships of Relationships
- Relationships of Relationships of Relationships

etc...

NGSI-LD Properties: Data Model

The Entity	Example	Notes
Has an **`id`**	**`urn:ngsi-ld:Building:store001`**	URI/URN. **`id`** must be unique.
Has a **`type`**.	**`https://uri.fiware.org/ns/data-models#Building`**	● Fully qualified URI of a well defined data model ● Short-hand strings for types, mapped to fully qualified URIs through the JSON-LD **@context.**
Has a series of properties	name, address, category etc.	This can be expanded into **`http://schema.org/address`**, which is known as a fully qualified name (FQN).
Has a series of properties-of-properties	a verified field for the address	This is the equivalent of NGSI v2 metadata
Has a series of relationships	managedBy	The **`object`** corresponds to the URI/URN of another data entity. Equivalent of NGSI v2 **`refXXX`**
Has a series of properties-of-relationships	managedBy.since	Holds additional information about a relationship. This is the equivalent of metadata about a **`refXXX`** property
Has a series of relationships-of-relationships	managedBy.subordinateTo	holds the URI/URN of another relationship.

NGSI-LD Properties: Reading Entity Data as JSON-LD

NGSI-LD

```
curl -G -X GET \
  'http://localhost:1026/ngsi-ld/v1/entities' \
    -H 'Link: <https://fiware.github.io/data-models/context.jsonld>;
    rel="http://www.w3.org/ns/json-ld#context";
    type="application/ld+json"' \
    -H 'Accept: application/ld+json' \
    -d 'type=Building' \
    -d 'options=keyValues'
```

- Response is just a JSON payload plus an `@context`
- `@context` can be passed either in the `Link` header or the payload body:
  - `Accept: application/ld+json` to include the `@context` as a JSON attribute
  - `Accept: application/json` returns plain old JSON objects - `@context` is passed as a Link header

```
[
  {
    "@context": "https://fiware.github.io/data-models/context.jsonld",
    "id": "urn:ngsi-ld:Building:store001",  "type": "Building",
    "address": {
       "streetAddress": "Bornholmer Straße 65",
       "addressRegion": "Berlin",
       "addressLocality": "Prenzlauer Berg",
       "postalCode": "10439"
    },
    "name": "Bösebrücke Einkauf",
    "category": “commercial",
    "location": {
       "type": "Point", "coordinates": [13.3986, 52.5547]
    }
  }
]
```

NGSI-LD Properties: What to call a location?

```
"location": {
    "type": "GeoProperty",
    "value": {
        "type": "Point",
        "coordinates": [13.3986, 52.5547]}
    }
}
```

- place ?
- locatedAt ?
- geocoordinate ?
- geocoordinates ?
- ubicación ?
- standort ?
- 置き場所 ?
- location ✓

NGSI-LD core @context

```
"@context": {
  "ngsi-ld": "https://uri.etsi.org/ngsi-ld/",
  "geojson": "https://purl.org/geojson/vocab#",
  "id": "@id",
  "type": "@type",
  "value": "ngsi-ld:hasValue",
... etc.
  "GeoProperty": "ngsi-ld:GeoProperty",
  "Point": "geojson:Point",
  "coordinates": {
      "@container": "@list",
      "@id": "geojson:coordinates"
  },
  "location": "https://uri.etsi.org/ngsi-ld/location",
... etc.
}
```

With NGSI-LD core **@context** a location is **always** `https://uri.etsi.org/ngsi-ld/location`
Thereafter, with JSON-LD you **may** map your preferred short name if necessary

NGSI-LD Relationships: Traversing Edge Nodes

Creating proper machine-readable Linked Data is **fundamental** to NGSI-LD.

From: https://www.w3.org/TR/json-ld/#dfn-graph

A JSON-LD document serializes a dataset which is a collection of graphs
A graph is a labeled directed graph, i.e., a set of nodes connected by edges.

In NGSI-LD:

- Node = NGSI Entity
- Edge = A relationship attribute linking two NGSI Entities

Therefore NGSI Linked Data relies on three separate definitions:

1. A definition that a particular attribute within an NGSI entity really represents a link
2. A machine readable definition of that link in the Data Model (i.e. the `@context`)
3. A machine readable definition of the set of all types of links available (the `@graph`)

NGSI-LD Relationships: 1. Creating Entities

Relationship Links within an NGSI Entity are formally defined using:

`"type": "Relationship"` OR `"@type": "https://uri.etsi.org/ngsi-ld/Relationship"`

The attribute of the linked entity is an `object` rather than a `value`

```
curl -X POST \
  http://localhost:1026/ngsi-ld/v1/entities/urn:ngsi-ld:Shelf:unit001/attrs \
  -H 'Content-Type: application/ld+json' \
  -H 'fiware-servicepath: /' \
  -d '{
    "stocks": { "type": "Relationship","object": "urn:ngsi-ld:Product:001"},
    "numberOfItems": {"type": "Property","value": 50},
    "locatedIn" : {
         "type": "Relationship", "object": "urn:ngsi-ld:Building:store001",
         "requestedBy": {"type": "Relationship","object": "urn:ngsi-ld:Person:bob-the-manager"},
         "installedBy": {"type": "Relationship","object": "urn:ngsi-ld:Person:employee001"},
         "statusOfWork": {"type": "Property","value": "completed"}
    },
    "@context": "https://fiware.github.io/tutorials.Step-by-Step/tutorials-context.jsonld"
}'
```

NGSI-LD Relationships: 2. Machine Readable Data Models

For the simplified JSON-LD output, relationship links within the `@context` can be formally defined using: `"@type": "@id"`

FIWARE Data Models @context

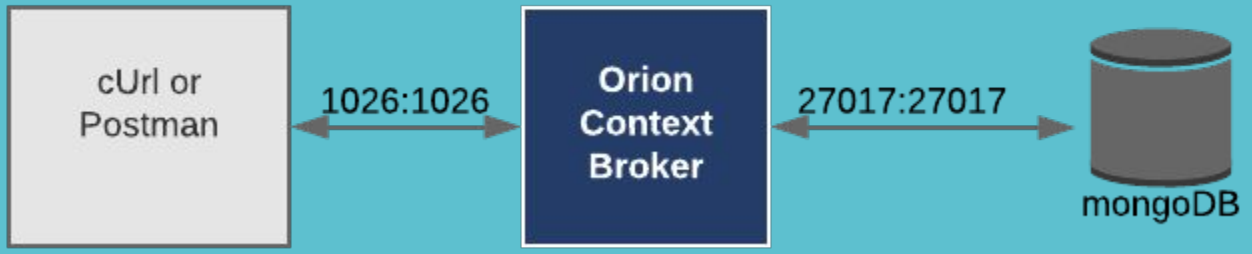
```
    "@context": {
      "tutorial": "https://fiware.github.io/tutorials.Step-by-Step/schema/",
      "Product": "tutorial:Product",
      "Shelf": "tutorial:Shelf",
    ...etc
      "installedBy": {
        "@id": "tutorial:installedBy",
        "@type": "@id"
      },
      "requestedBy": {
        "@id": "tutorial:requestedBy",
        "@type": "@id"
      },
    ...etc
```

NGSI-LD Relationships: 3. Machine Readable Links

FIWARE Data Models @graph

```
"@graph": [
   {
     "@id": "tutorial:Product",
     "@type": "rdfs:Class",
     "rdfs:comment": [
         {"@language": "en", "@value": "Product is sold in a Store."},
         {"@language": "ja", "@value": "製品はストアで販売されている物"}],
     "rdfs:label": [{"@language": "en", "@value": "Product"}, {"@language": "ja", "@value": "製品"}],
     "rdfs:subClassOf": {"@id": "http://schema.org/Thing"}
   },
… etc
  {
     "@id": "tutorial:requestedBy",
     "@type": "https://uri.etsi.org/ngsi-ld/Relationship",
     "schema:domainIncludes": [{"@id": "tutorial:Shelf"}, {"@id": "tutorial:StockOrder"}],
     "schema:rangeIncludes": [{"@id": "schema:Person"}],
     "rdfs:comment": [
         {"@language": "en","@value": "Object requested by person."},
         {"@language": "ja","@value": "人が要求したオブジェクト"}],
     "rdfs:label": [{"@language": "en", "@value": "requested by"},{"@language": "ja", "@value": "要求者"}]
  },
]
```

Demo: NGSI-LD - Relationships



NGSI-LD Subscriptions: Creating a Subscription

NGSI-LD

```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/subscriptions/' \
-H 'Content-Type: application/ld+json' \
--data-raw '{
   "description": "Notify me of low stock in Store 001",
   "type": "Subscription",
   "entities": [{"type": "Shelf"}],
   "watchedAttributes": ["numberOfItems"],
   "q": "numberOfItems<10;locatedIn==urn:ngsi-ld:Building:store001",
   "notification": {
      "attributes": [ "numberOfItems", "stocks","locatedIn"],
      "format": "keyValues",
      "endpoint": {
         "uri": "http://tutorial:3000/subscription/low-stock-store001",
         "accept": "application/json"
      }
   },
   "@context":
   "https://fiware.github.io/tutorials.Step-by-Step/tutorials-context.jsonld"
}'
```

Sample Key-Values Payload

```
{
 "id": "urn:ngsi-ld:Notification:60812d06f2ebd727e1c425a8",
 "type": "Notification",
 "subscriptionId":
   "urn:ngsi-ld:Subscription:60812c7bf2ebd727e1c425a4",
 "notifiedAt": "2021-04-22T08:00:06.741Z",
 "data": [
  {
   "id": "urn:ngsi-ld:Shelf:unit001",
   "type": "Shelf",
   "locatedIn": "urn:ngsi-ld:Building:store001",
   "numberOfItems": 8,
   "stocks": "urn:ngsi-ld:Product:001"
  }
 ]
}
```

NGSI-LD Registrations: Creating a Registration

NGSI LD

```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/csourceRegistrations/' \
-H 'Content-Type: application/json' \
-H 'Link: <https://fiware.github.io/tutorials.Step-by-Step/tutorials-context.jsonld>;
   rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"' \
--data-raw ' {
   "type": "ContextSourceRegistration",
   "mode" : "exclusive",
   "operations" : "retrieveOps",
   "information": [
      {
         "entities": [
            {"type": "Building",  "id": "urn:ngsi-ld:Building:store001"}
         ],
         "propertiesNames": [
            "tweets"
         ]
      }
   ],
   "endpoint": "http://context-provider:3000/static/tweets"
}'
```

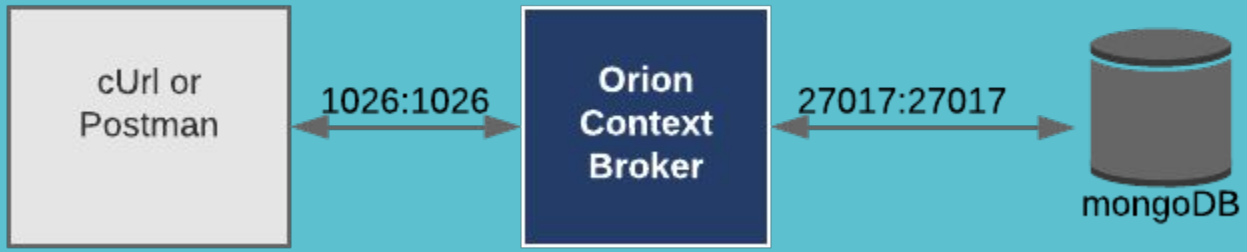
Note that `properties` was defined in the 1.1.1 NGSI-LD core context

Since 1.3.1, `properties` has been replaced with two separate attributes - `propertyNames` and `relationshipNames` - this change has been made in order to offer full GeoJSON-LD support.

In 1.6.1 four different modes of Registration are now defined. `inclusive`, `exclusive`, `redirect`, `auxiliary`

1.6.1 also groups operations into groups e.g. `federationOps`, `retrieveOps`

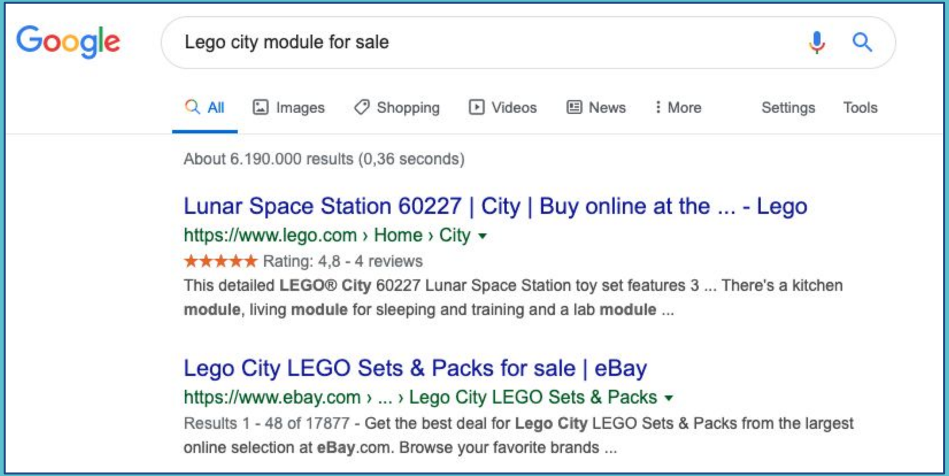
Demo: NGSI-LD - Subscriptions and Registrations



Context Data as Linked Data - How does it help? **Data Sharing**

Rich Text Snippets

Standard `schema.org/Product` data model marked up as JSON-LD on the web. Interpreted by third parties. Search Engine can display product rating on screen. System "knows" if a product is out of stock.



About 6.190.000 results (0,36 seconds)

Lunar Space Station 60227 | City | Buy online at the ... - Lego
https://www.lego.com › Home › City
★★★★★ Rating: 4,8 - 4 reviews
This detailed **LEGO® City** 60227 Lunar Space Station toy set features 3 ... There's a kitchen **module**, living **module** for sleeping and training and a lab **module** ...

Lego City LEGO Sets & Packs for sale | eBay
https://www.ebay.com › ... › Lego City LEGO Sets & Packs
Results 1 - 48 of 17877 - Get the best deal for **Lego City** LEGO Sets & Packs from the largest online selection at **eBay**.com. Browse your favorite brands ...

NGSI-LD Supermarket Tutorial

Third party ARV could "know" when a shelf needs filling and retrieve goods from the warehouse

No need to reprogram for new customers if data follows the `fiware.org/ns/data-models`, or the JSON-LD can be converted to do so.

FIWARE

FIWARE in Production

NGINX

HAPROXY

OPENSHIFT

FIWARE

Requirements

- expected load and load-behaviour
  - is growth expected?
  - stable load vs. high/low-scenarios
- availability, acceptable downtimes
- latency (per service)
- security and privacy
  - encryption at rest/ in transition?
  - GDPR requirements?
- costs - how much can we pay for the system?

Why Kubernetes?

- standardized abstraction layer and orchestration-tool
- support automated configurations and deployments
- built in mechanisms to fulfill scale and availability requirements
- broad support for operational tooling, especially:
  - logging
  - monitoring
  - alerting

Standardized abstraction layer

- allows to use the same recipes on different environments
    - Helm-Charts: FIWARE/helm-charts
    - production-grade recipes available for multiple components
- available on different infrastructure
    - managed versions on different Cloud/Infrastructure Providers
    - self-managed On-Premise
    - different distributions available, f.e. RedHat OpenShift

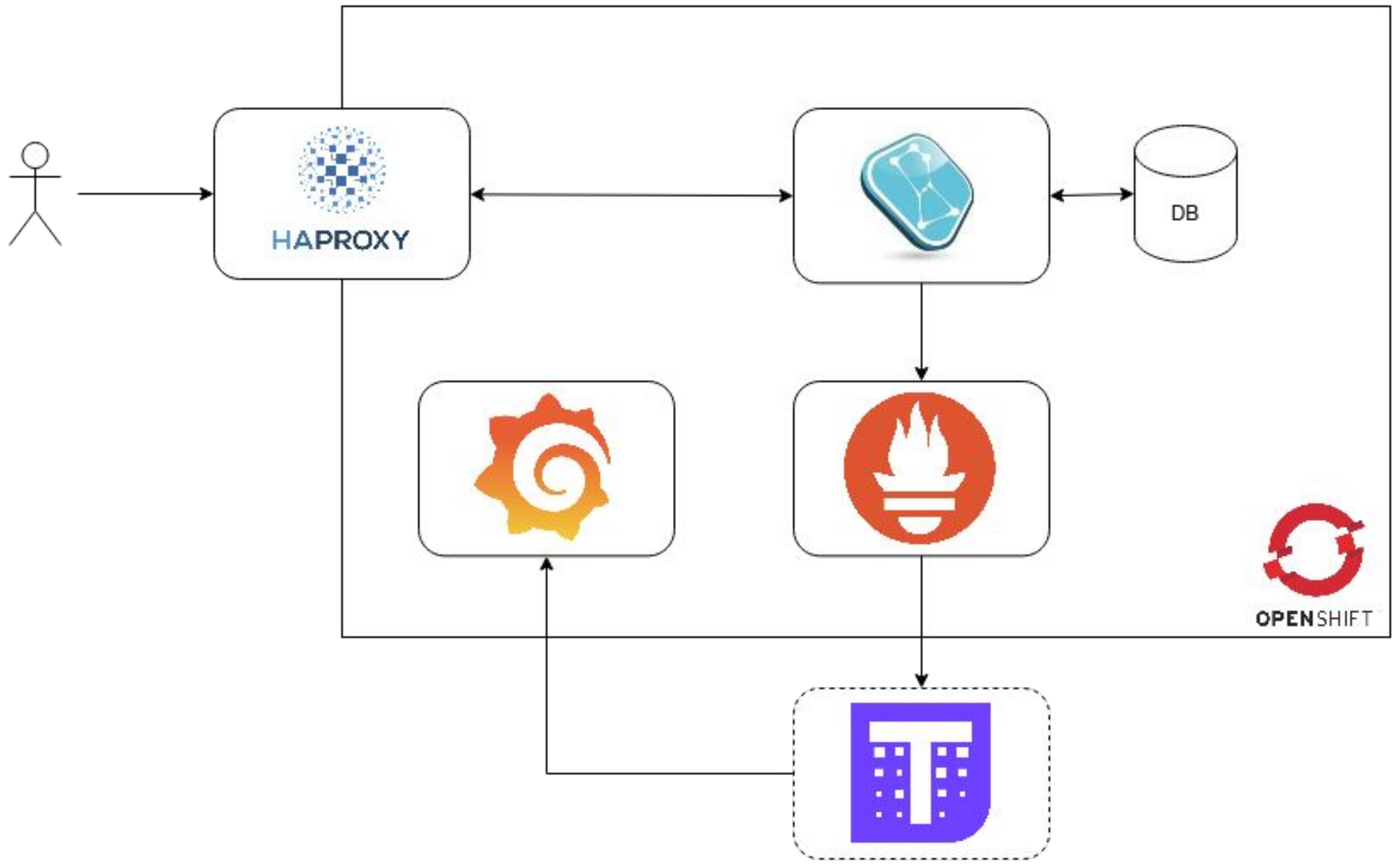
Automated configuration and deployment

- Helm or Kustomize for configurable deployments
- Operators for automating update/scaling/etc.
- GitOps tooling for managing deployments, f.e.
  - ArgoCD
  - Flux

Mechanisms for Scaling and Availability

- nativ support for horizontal scaling through:
    - ReplicaSets
    - StatefulSets
- automatically distribute workloads through different availability zones:
    - node-labeling/pooling
    - affinity/anti-affinity
    - tolerations
- support for update-strategies

Overview



FIWARE

Practical examples

- Orion-LD on FIWARE-Ops/fiware-gitops
- Running at scale: FIWARE/load-tests
- Get your feets wet with Kubernetes:
  - fiware-on-k3s
  - OpenShift Sandbox

FIWARE
Global
Summit

Thanks!

Gran Canaria, Spain
14-15 September, 2022
#FIWARESummit

FIWARE
Open APIs for Open Minds