



OCCLware : Year 1 Milestone

Docker Studio, Studio Factory, pluggable XaaS runtime
& Linked Data use case end-to-end POC

Marc Dutoo, Open Wide / Smile

Cloud Expo Europe, London

Open Cloud Developer Park - April, 13rd 2016

Overview

Speaker

- Marc Dutoo, Head of R&D Dept. at Open Wide, a Smile group company
 - OCCIware coordinator, Data / Cloud expert

Schedule

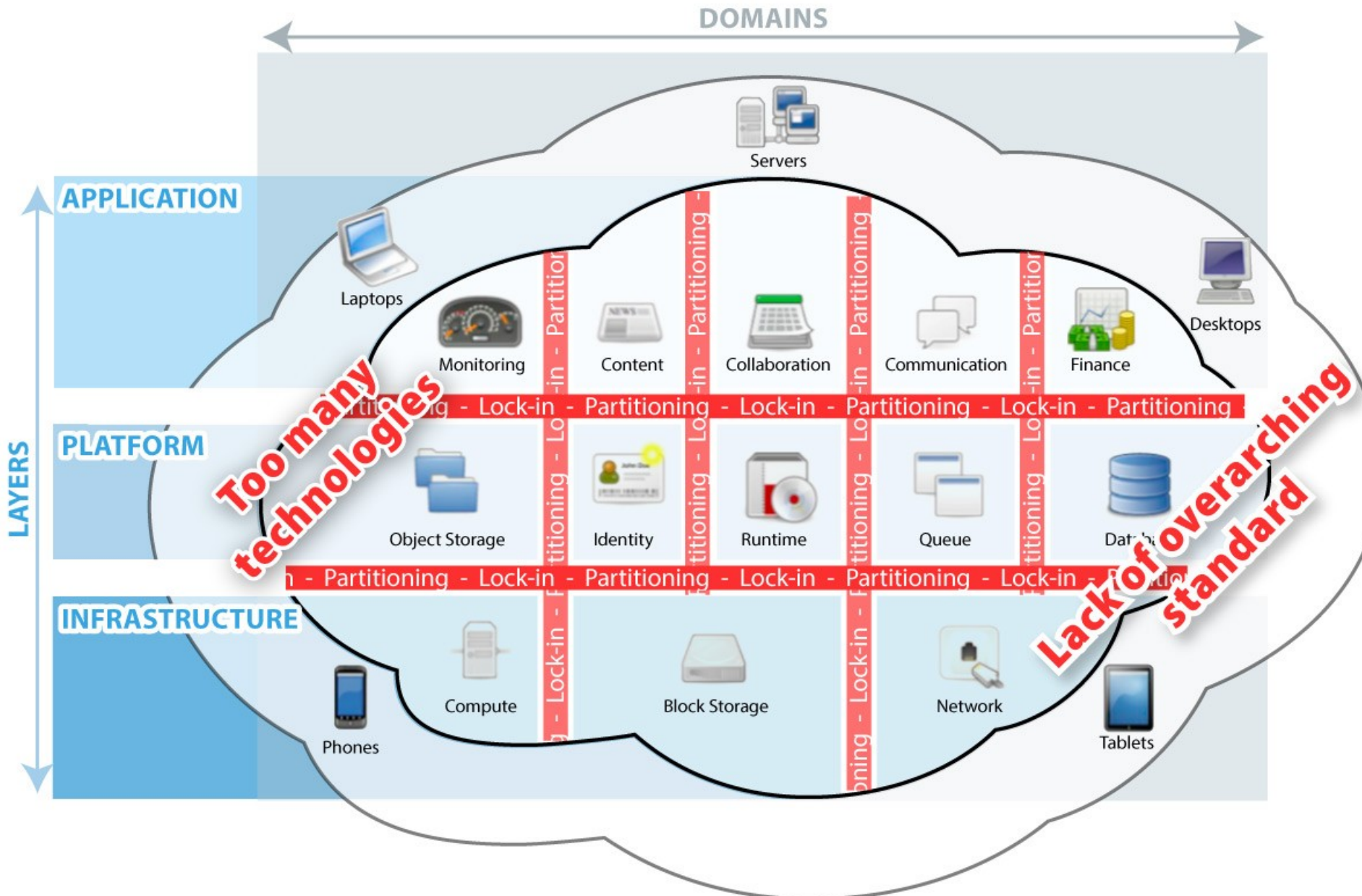
- 6' OCCI(ware) introduction
- 7' OCCIware Year 1 Outputs
- 7' LDaaS end-to-end proof of concept with erocci



OCCIware Factsheet

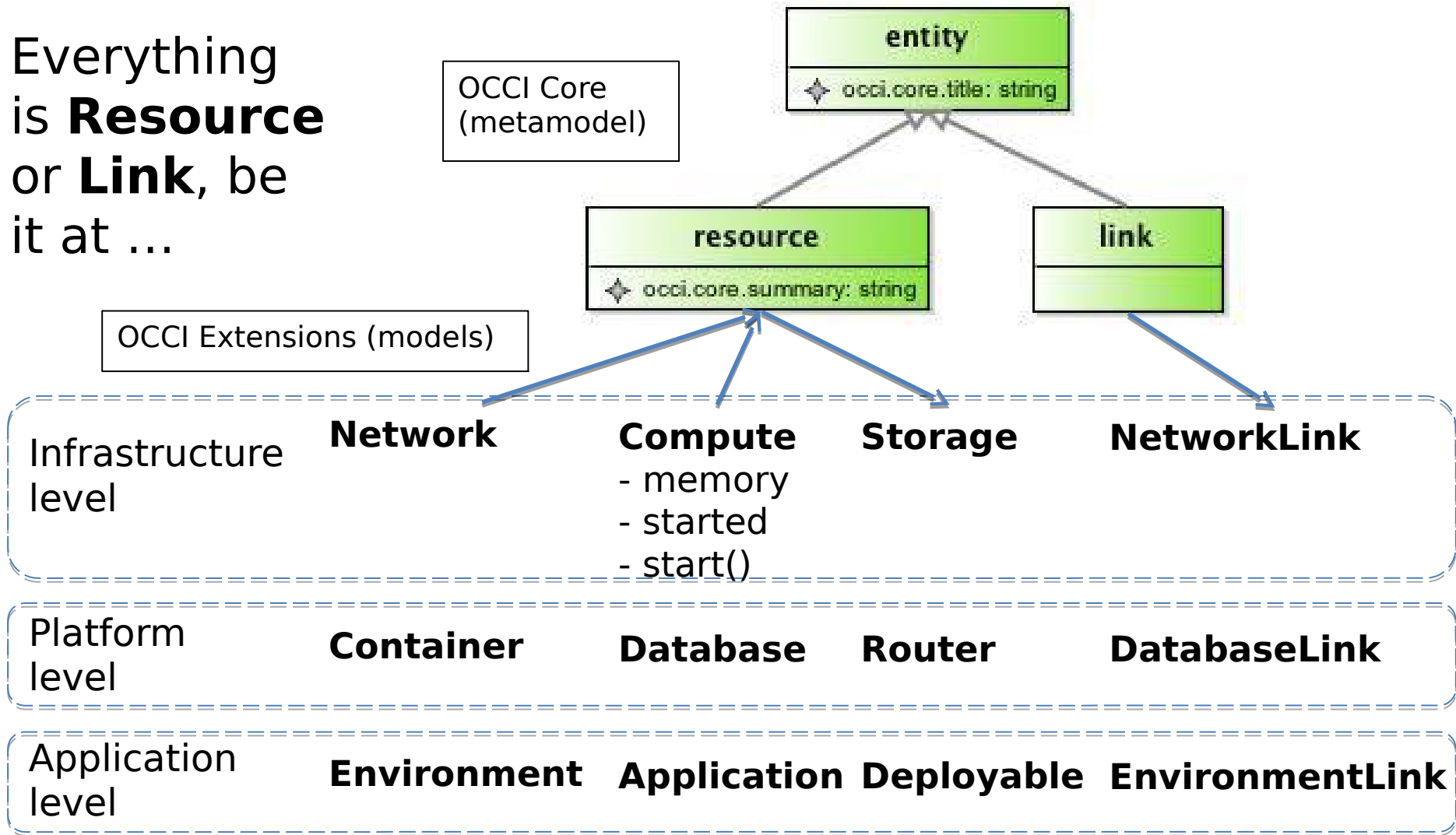
- 72 man year, 5,6m€ budget, sponsored by French ministry of Industry over 2015-2018
- 3 academics, 5 companies, 2 associations
- To lower Cloud Computing adoption costs and **break up barriers** between its various implementations, layers, domains
 - Especially Data Center, deployment, Big Data, Linked Data
- By bringing to OGF's Open Cloud Computing Interface (**OCCI**) the power of **formal** languages and **model** driven engineering (MDE)

Cloud Computing - the problem



OCCI 101

Everything is **Resource** or **Link**, be it at ...



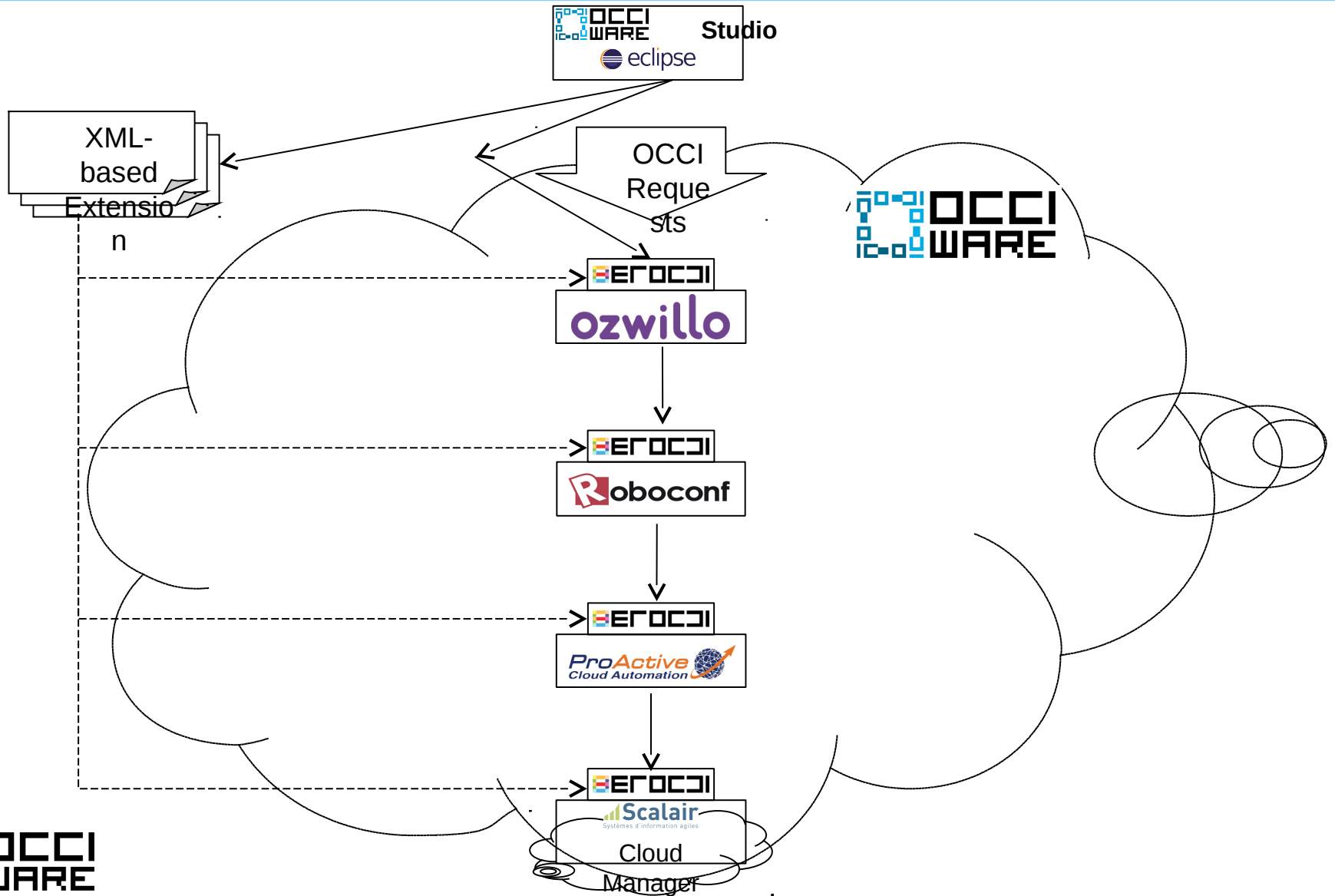
OCCIware Target Outputs

- A formal, model-driven platform to manage any Cloud resource
 - Formal model of OCCl, on MIT's Alloy - **Inria & TSP**
 - OCClware Studio, on Eclipse EMF & Sirius - **Obeo**
 - OCClware@Runtime & console, using Models@Runtime and erocci
 - **ActiveEon & Scalair**
 - Deploy@OCCIware - **ActiveEon & UJF**
- 4 Use cases
 - Data Center as a Service / IaaS, on **Scalair** infrastructure
 - Big Data / HPC, on **ActiveEon** ProActive HPC platform
 - Linked Open Data, on Ozwillo app store's Datacore - **Pôle Numérique & Open Wide**
 - Deployment interoperability, on **Linagora & ActiveEon's**
- Open Source (**OW2**, Eclipse) and standardization (OGF) with help from a 10-strong international **Scientific Orientation Committee** towards an international standard

OCCLware Year 1 Main Outputs

- OCCLware Studio
 - **OCCLware Studio Factory** : produce **visually customizable** diagram editors for any Cloud configuration business domain modeled in OCCl using the OCCl Extension Studio
 - such as the flagship **Docker Studio**
 - Models@Runtime : e.g. **deploy** your Docker diagram by a mere click in the Docker Studio and see the result there
- OCCLware runtime
 - erocci, a scalable generic **OCCl bus** written in erlang
 - that federates multiple Cloud runtimes ("**backends**") using e.g. **Java** through Dbus or **python**,
 - such as the Roboconf **PaaS server** and the ActiveEon Cloud Automation **multi-iaaS connector**

Big Picture – Studio to Runtime to all providers



OCCIware generic Studio

- Generate doc, runtime configuration, scripts...

The screenshot displays the OCCiware generic Studio interface, which is a modeling environment for OCCi (Open Cloud Connector Interface) infrastructure. The interface is divided into several panes:

- Model Explorer:** Shows the project structure, including 'infrastructure' and 'infrastructure.occie'. The 'infrastructure.occie' package is expanded, showing various classes and mixins.
- Code Editor:** Displays the source code for the 'infrastructure.occie' package. The code defines several classes and mixins, including 'storageLink', 'networkInterface', 'ipnetwork', 'os_tpl', 'ipnetworkinterface', 'os_tpl', 'ssh_key', and 'user_data'. Each class and mixin is annotated with OCCi metadata, such as titles and attributes.
- UML Class Diagram:** Shows the relationships between the classes and mixins. The diagram includes a 'core' package with 'entity' and 'link' classes. 'entity' is the base class for 'resource' and 'link'. 'resource' is the base class for 'network', 'compute', and 'storage'. 'link' is the base class for 'networkInterface', 'storageLink', and 'ipnetwork'. The diagram also shows 'os_tpl' and 'ssh_key' as subclasses of 'compute', and 'user_data' as a subclass of 'compute'. The diagram also shows 'StorageStatus' and 'StorageLinkStatus' as subclasses of 'StorageStatus' and 'StorageLinkStatus' respectively.
- Properties:** Shows the properties of the selected class, including 'Name' and 'Scheme'.
- Extension infrastructure:** Shows the extension infrastructure, including 'Rulers & Grid', 'Appearance', and 'Semantic'.

```
classDiagram
    class core {
        class entity {
            +String occi_core_title
        }
        class link
    }
    class infrastructure {
        class resource {
            +String occi_core_summary
        }
        class network {
            +Vlan an
            +Token label
            +NetworkStatus inactive
            +String state_message
        }
        class compute {
            +Architecture occi_compute_architecture
            +Core occi_compute_cores
            +String occi_compute_hostname
            +Share occi_compute_share
            +GHz occi_compute_speed
            +GIB occi_compute_memory
            +ComputeStatus inactive
            +String occi_compute_state_message
            +start()
            +stop()
            +restart()
            +suspend()
            +save()
        }
        class storage {
            +GIB occi_storage_size
            +StorageStatus occi_storage_state
            +String occi_storage_state_message
            +online()
            +offline()
        }
        class networkInterface {
            +String occi_networkinterface_interface
            +Mac occi_networkinterface_mac
            +NetworkInterfaceState occi_networkinterface_state
            +String occi_networkinterface_state_message
        }
        class storageLink {
            +String occi_storageLink_deviceid
            +String occi_storageLink_mountpoint
            +StorageLinkStatus occi_storageLink_state
            +String occi_storageLink_state_message
        }
        class ipnetwork {
            +IpAddressRange an
            +IpAddressRange gateway
            +Allocation occi_network_allocation
        }
        class os_tpl {
            +Architecture
            +ComputeStatus
        }
        class ssh_key {
            +String occi_credentials_ssh_publickey
        }
        class user_data {
            +String occi_compute_userdata
        }
        class StorageStatus {
            +online
            +inactive
            +error
        }
        class StorageLinkStatus {
            +active
            +inactive
            +error
        }
        class ipnetworkInterface {
            +String occi_networkinterface_in
            +String occi_networkinterface_m
            +String occi_networkinterface_st
            +String occi_networkinterface_st
        }
    }
    core.entity <|-- infrastructure.resource
    core.entity <|-- infrastructure.link
    infrastructure.resource <|-- infrastructure.network
    infrastructure.resource <|-- infrastructure.compute
    infrastructure.resource <|-- infrastructure.storage
    infrastructure.link <|-- infrastructure.networkInterface
    infrastructure.link <|-- infrastructure.storageLink
    infrastructure.link <|-- infrastructure.ipnetwork
    infrastructure.compute <|-- infrastructure.os_tpl
    infrastructure.compute <|-- infrastructure.ssh_key
    infrastructure.compute <|-- infrastructure.user_data
    infrastructure.compute <|-- infrastructure.StorageStatus
    infrastructure.compute <|-- infrastructure.StorageLinkStatus
    infrastructure.compute <|-- infrastructure.ipnetworkInterface
```

Docker Studio

• From red to green : let your models go live !

The screenshot displays the Eclipse IDE interface with the Docker Studio extension. The main workspace shows a network diagram titled "Storm-topology" with a green background, indicating it is running. The diagram consists of several nodes (green cubes) connected by blue arrows, representing a complex network topology. The nodes are labeled: "ranking-bolt" (top), "rolling-count-bolt-1" and "rolling-count-bolt-2" (middle), "count-bolt-1", "count-bolt-2", and "count-bolt-3" (bottom), and "word-spout-2" and "word-spout-1" (bottom-most). To the left of the diagram are three red boxes labeled "default", "bingo", and "spinoz-vm".

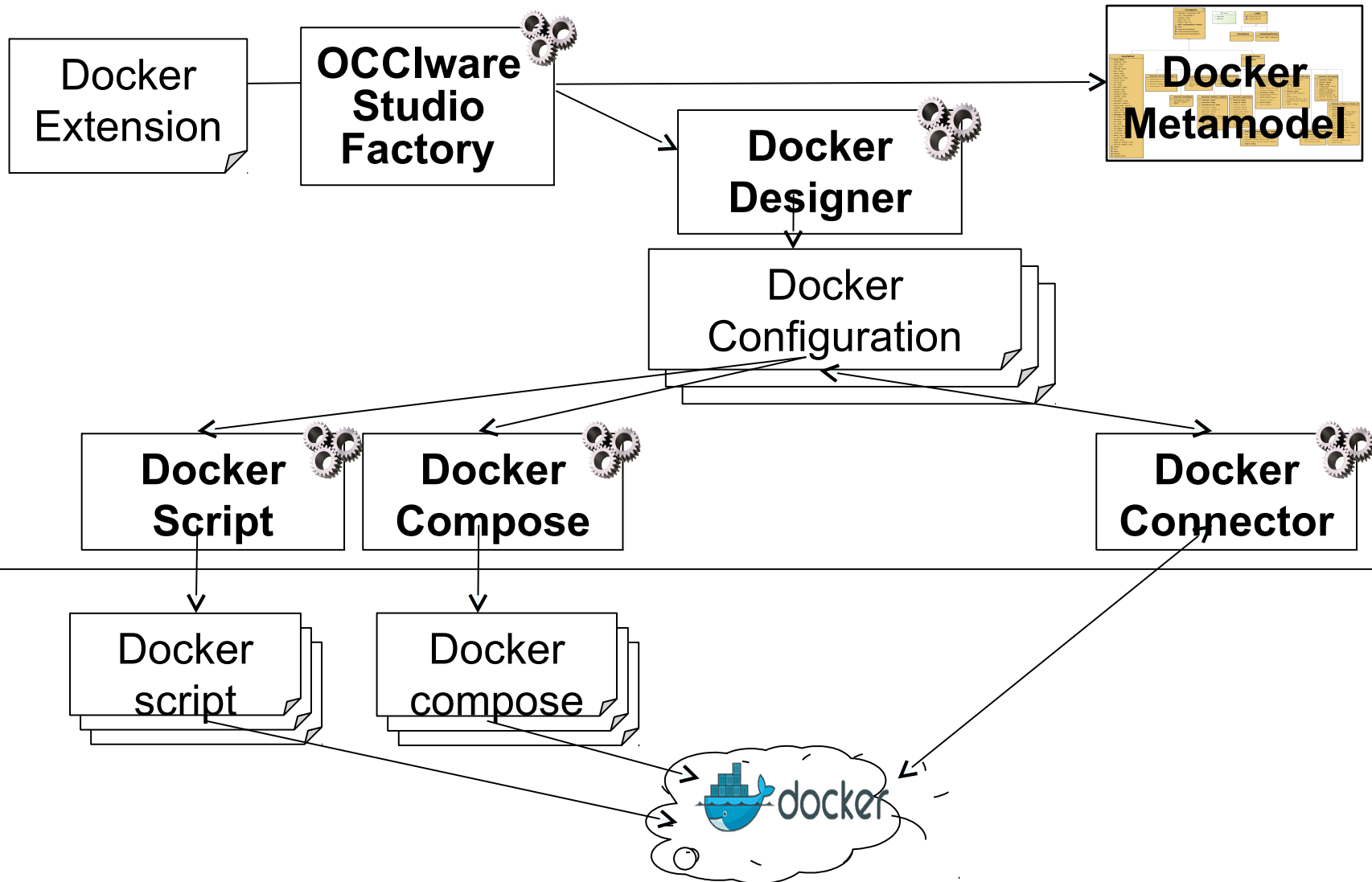
The left sidebar shows the "Model Explorer" with a tree view of the project structure:

- docker-modeler
 - Project Dependencies
 - new_configuration.occic
 - Configuration
 - new Docker configuration
 - Machine Virtual Box mymachine
 - Container mycontainer
 - Machine Virtual Box template
 - Machine VMware Fusion abikou
 - Machine Virtual Box test
 - Machine Virtual Box dev
 - Machine Virtual Box home-automation
 - Machine Open Stack occiware
 - Machine Virtual Box dev1
 - Machine Virtual Box Storm-topolog
 - Machine Virtual Box default
 - Machine Virtual Box Storm-topology
 - Container word-spout-2
 - Container word-spout-1
 - Container count-bolt-1
 - Container count-bolt-2
 - Container count-bolt-3
 - Container rolling-count-bolt-1
 - Container rolling-count-bolt-2
 - Container ranking-bolt
 - Machine Virtual Box bingo
 - Machine VMware Fusion spinoz-vm

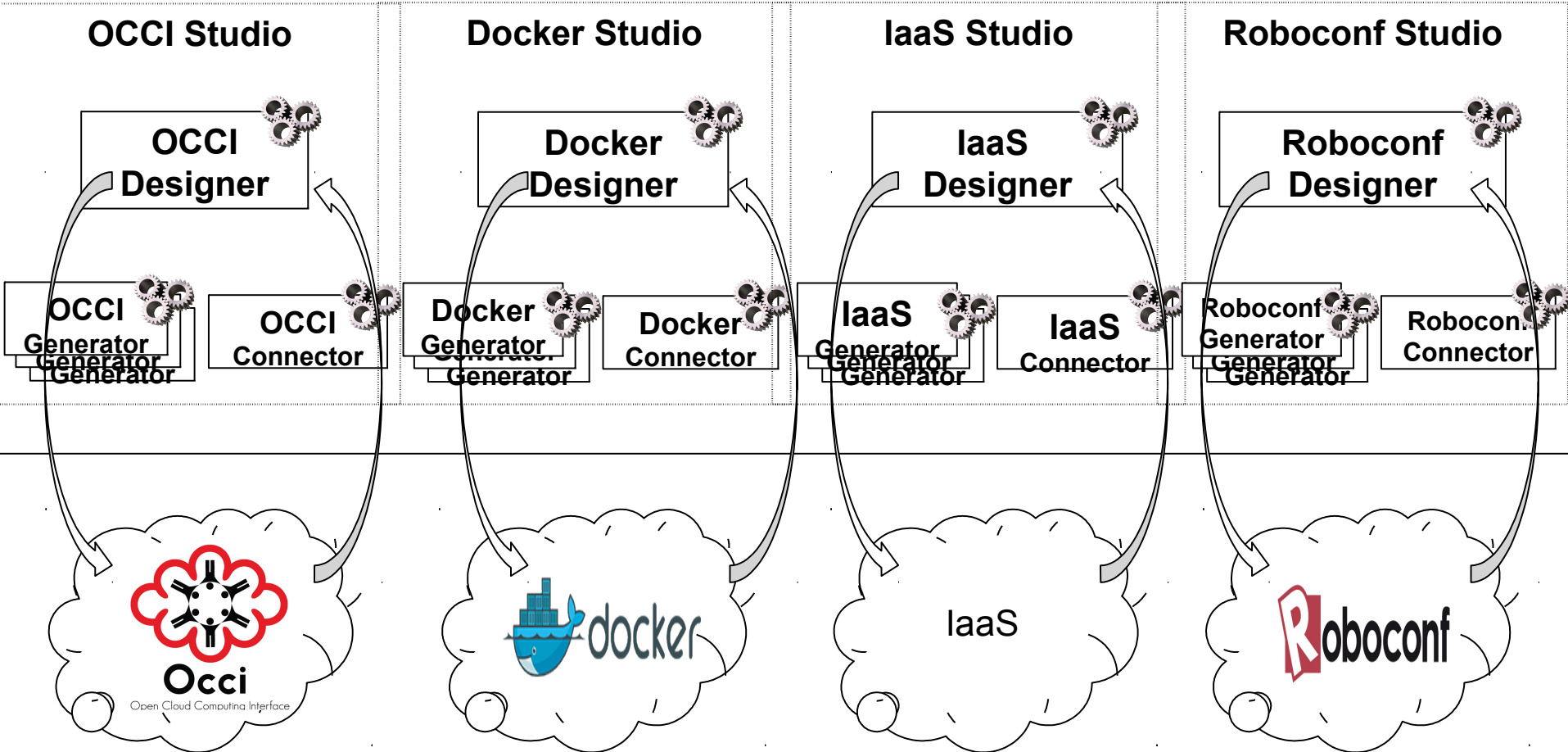
The bottom right panel shows the "Configuration" section with a table of properties:

Property	Value
Configuration	
Use	
Semantic	
Rulers & Grid	
Appearance	
Use	Extension docker

Docker Studio - features

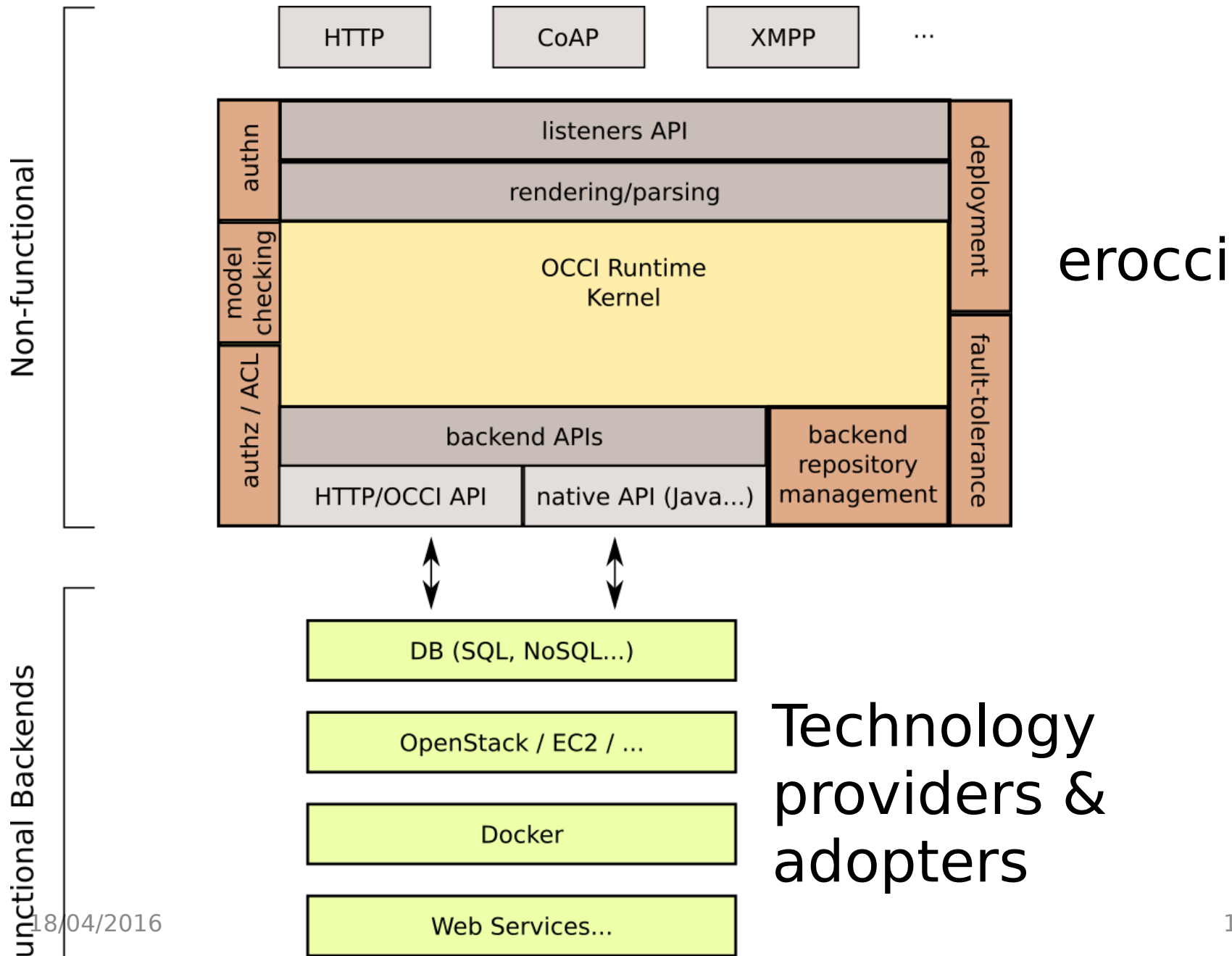


OCCIware Studio Factory

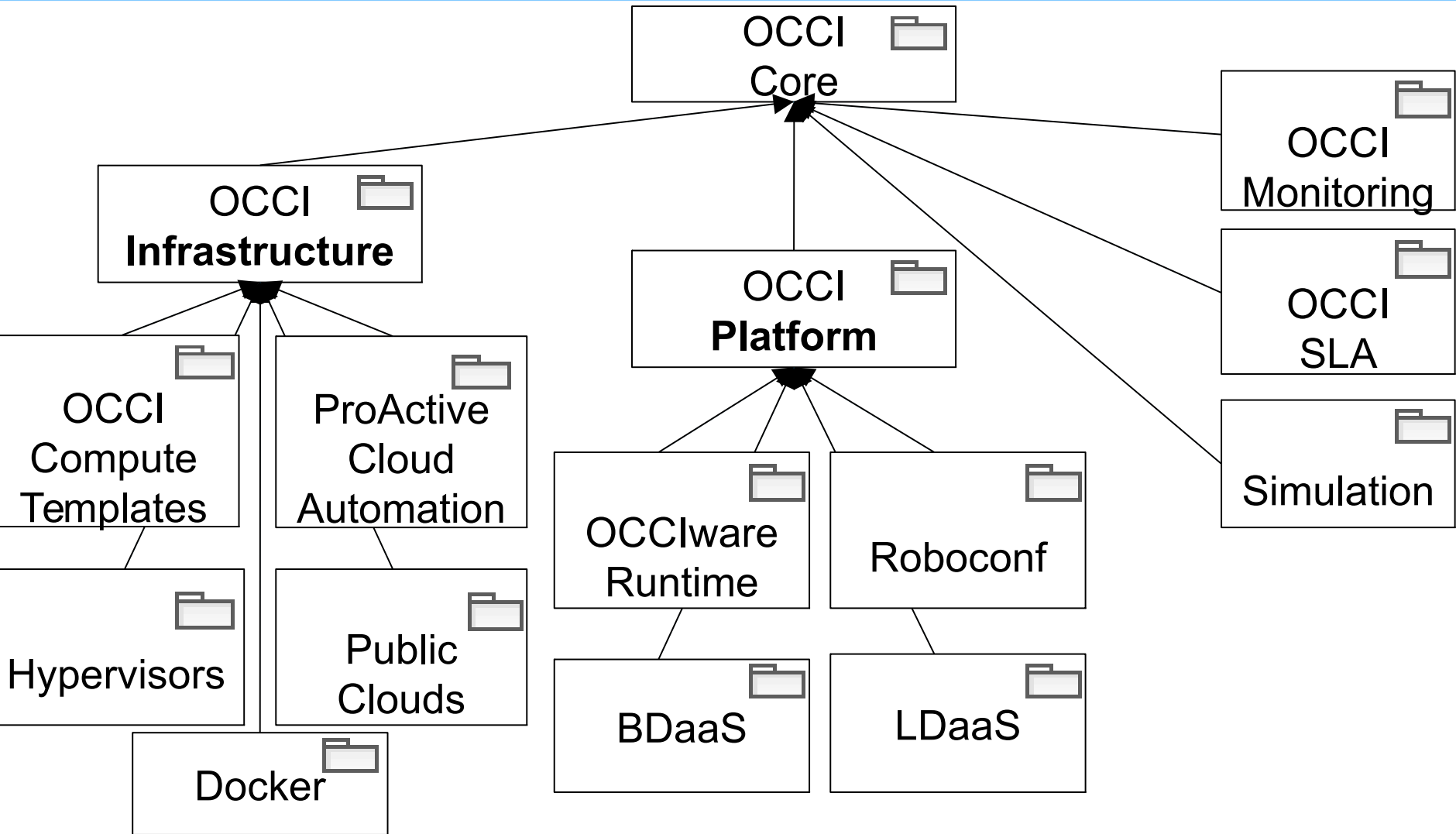


... your own Studio !

Runtime Architecture



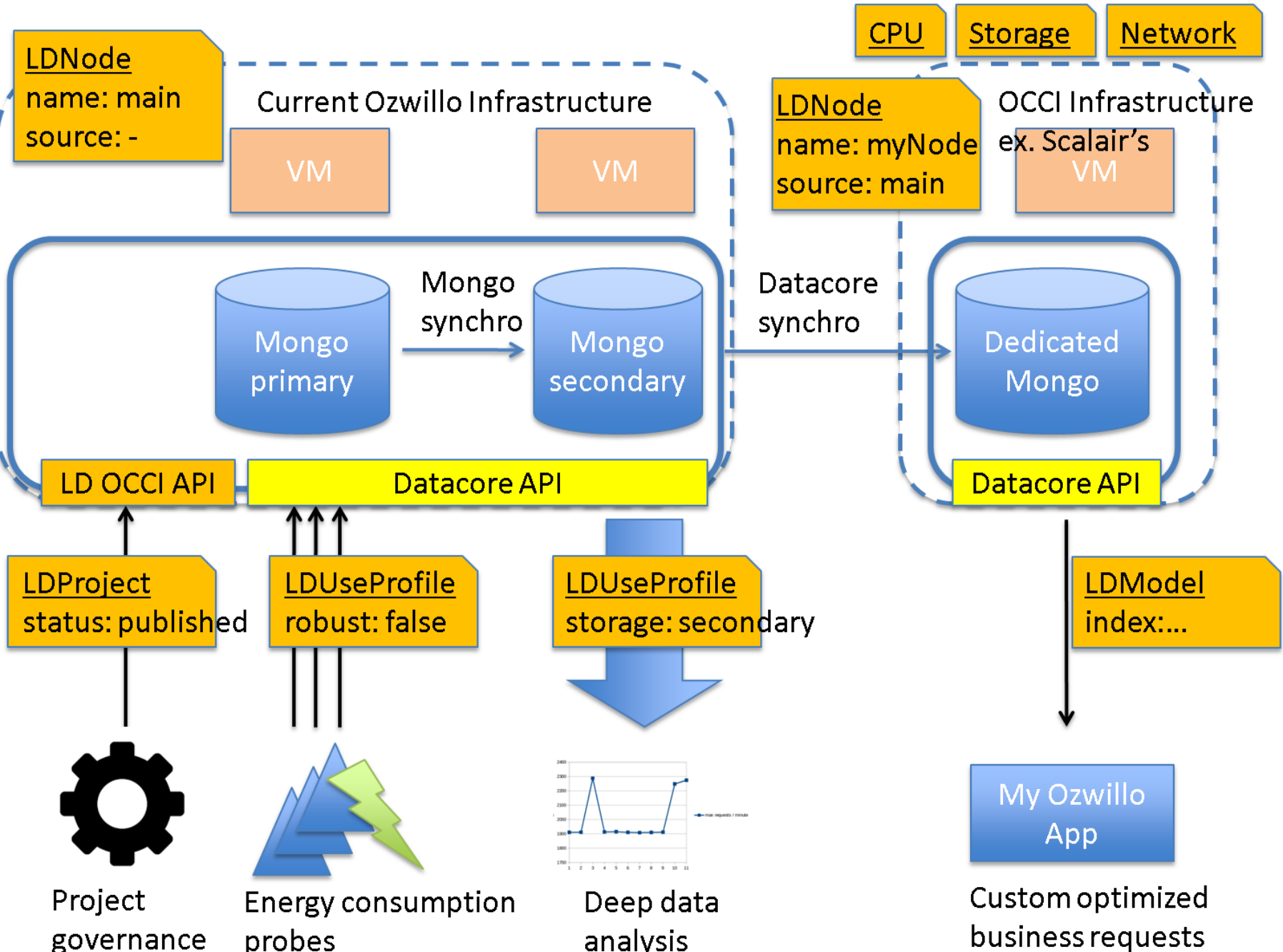
Supported OCCI Extensions



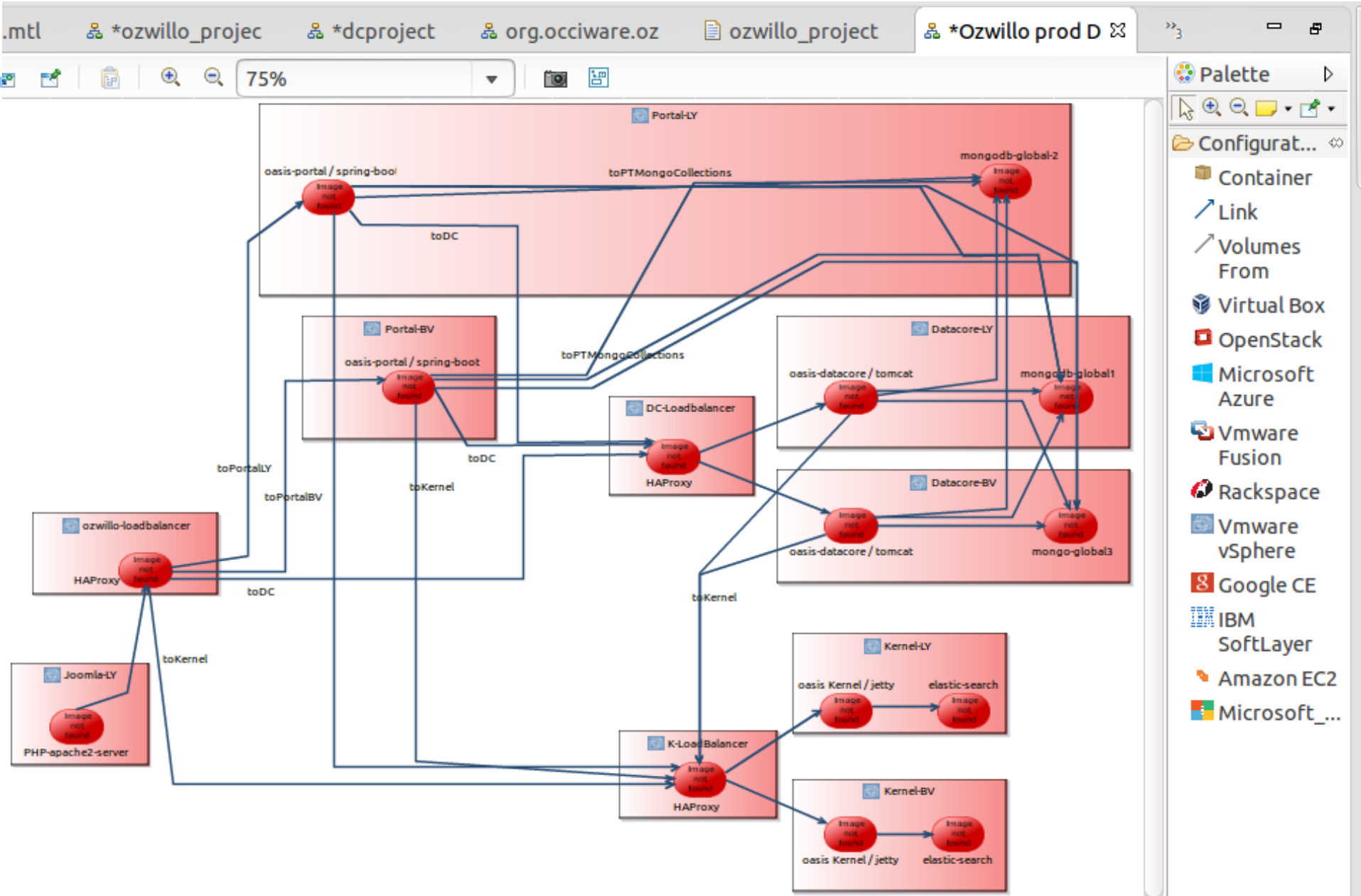
OCCI POC: Linked Open Data

- Linked Open Data ? That's Open Data sets that can be **cross-queried** because they have been reconciled together
- Enter Ozwillo Datacore - it holds data that is **shared between applications** of the Ozwillo app store : geographical elements, organizations, reusable app business data...
- Datacore OCCI(ware) use cases = Datacore « configuration as a Service » use cases : letting app developers configure...
 - its own data models, their rights and **governance** policy,
 - its own usage profiles : data collaboration app, **high-write** for IoT sensor status notifications, **analytics** for aggregating those
 - define custom indexes, up to deploy dedicated data **Caches** as a Service...
- => demo of OCCIware Studio & Runtime's first features
 - **demo 1** : try to **Dockerize** current Ozwillo production architecture
 - **demo 2** : **governance** of Linked Data projects (and their models)

Linked Data scenarii overview



D1 – Docker-specific Studio



D2 - Studio : OCCI extension

For governance of Linked Data projects & models

The screenshot displays the D2 Studio interface for an OCCI extension. The main workspace shows a class diagram for the 'core' package. The diagram includes the following classes and their attributes:

- String**: `java.lang.String`
- Boolean**: `boolean`
- Number**: `int`
- Version**: `long`
- List**: `java.util.List`
- Map**: `java.util.Map`
- entity**: `occi.core.title: String`
- resource**: `occi.core.summary: String`
- link**: (No attributes listed)
- project**: `name: String`, `localVisibleProjects: List`, `frozenModelNames: List`
- model**: `name: String`, `version: Version`, `majorVersion: Version`
- modelToProjectLink**: `alias: String`

The diagram shows inheritance relationships: **resource** and **link** inherit from **entity**. **project** and **model** inherit from **resource**. **modelToProjectLink** inherits from **link**.

The left sidebar shows the 'Model Explorer' with a tree view of the project structure, including folders like 'OCCI Configuration', 'model', and 'src-gen'. The bottom status bar shows a console message: `<terminated> GenOCCIPost ozwilllo [Acceleo Application] /usr/lib/jvm/jdk1.8.0_45/bin/java (13 nov. 2015 15:14:23)`.

...to doc & runtime configuration

*.occie extension file > Cloud Designer > Generate textile doc, erocci runtime XML conf...

The screenshot shows the Eclipse IDE interface. On the left, the Model Explorer displays a project structure with folders like 'model', 'src-gen', 'erocci', 'textile', and 'xml'. The file 'org.occiware.ozwillo.data.mo...' is selected under the 'textile' folder. On the right, the Textile Source editor shows the following content:

```
h1. OCCI Extension: org.occiware.ozwillo.data.models

| *Name* | org.occiware.ozwillo.data.models |
| *Scheme* | http://occiware.org/ozwillo/dcproject# |

h1. Imports

| *Name* | *Scheme* |
| <a href="core.textile">core</a> | http://schemas.org/occi/core# |

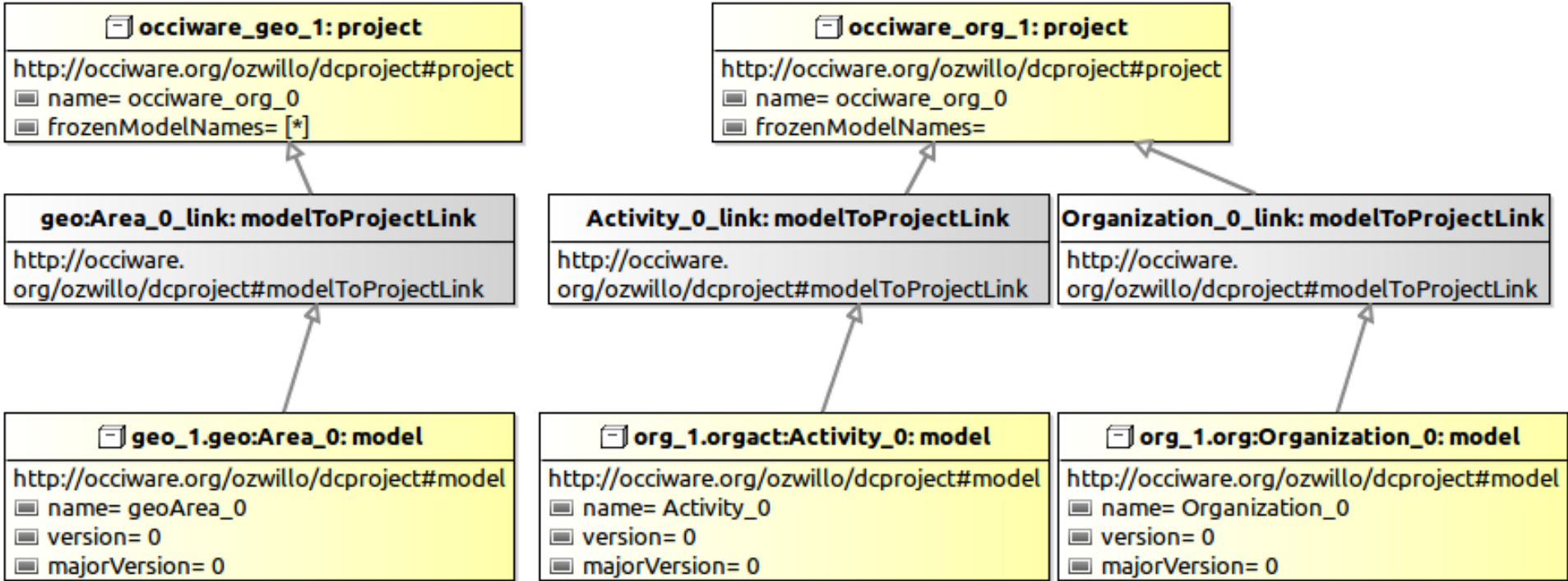
h1. Kinds

h2. Kind model

| *Term* | model |
| *Scheme* | http://occiware.org/ozwillo/dcproject# |
| *Title* | Model |
| *Parent* | <a href="core.textile#kind-resource">http://schemas.org/occi/core#resource</a> |
```

At the bottom of the editor, there are tabs for 'Textile Source' and 'Preview'.

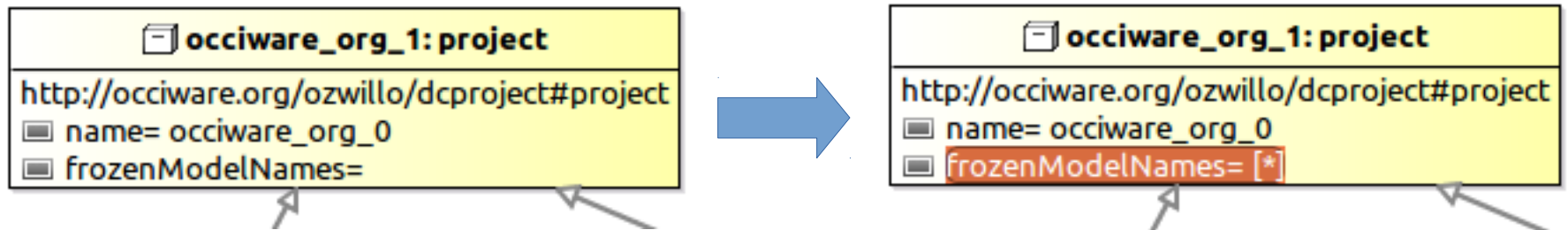
Studio : OCCI configuration



...to curl POST to runtime script

Governance action example :

in order to publish a stable version of the org_0 project, let's **freeze all its models** :



*.occic configuration file > Cloud Designer > Generate curl script :

```
curl $COPTS -X PUT $SERVER/project/occiware_geo_1 -H  
'Content-Type: text/occi' -H 'Category: project;  
scheme="http://occiware.org/ozwillo/dcproject#";  
class="kind";' -H 'X-OCCE-Attribute:  
name="occiware_org_0"' -H 'X-OCCE-Attribute:  
frozenModelNames=" [*] "'
```

Runtime setup (erocci with DBus)

- Configuration

- OCCI extension configuration : Core, **LDProject**
- **erocci backend : DBus, with the erocci-dbus-java bridge deployed along, which talks to the regular REST API of the Ozwillo Datacore**

```
vi sys.config
...
{erocci_backend_dbus,
 [{schemas, [
   {path, "/tmp/occi.xml"},
   {path, "/tmp/org.occiware.ozwillo.data.models.xml"}]]}]
},
...
```

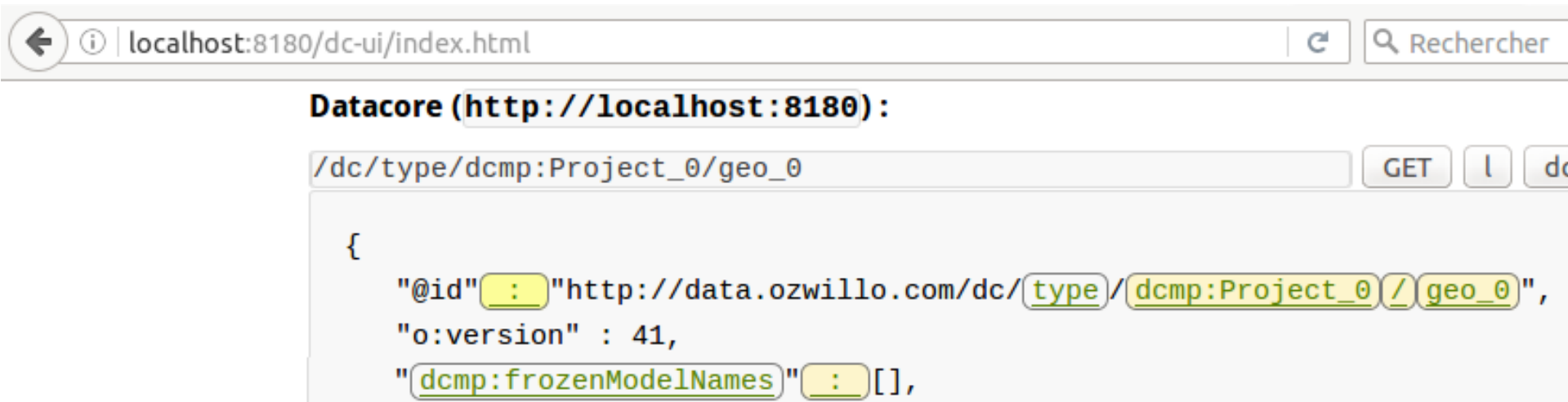
- Let's start it using **docker** :

```
sudo docker run --name="erocci_linked-data" -v
`pwd`/sys.config:/tmp/sys.config -v `pwd`/occi.xml:/tmp/occi.xml -v
`pwd`/org.occiware.ozwillo.data.models.xml:/tmp/org.occiware.ozwill
o.data.models.xml -P -t -i erocci/erocci
```



Initially, models are unfrozen

As shown in Linked Data server backoffice (API Playground) :



The screenshot shows a web browser window with the address bar containing `localhost:8180/dc-ui/index.html`. Below the browser, the API Playground interface is visible. The endpoint `/dc/type/dcmp:Project_0/geo_0` is selected, and the method `GET` is chosen. The response is a JSON object:

```
{
  "@id" : "http://data.ozwillo.com/dc/type/dcmp:Project_0/geo_0",
  "o:version" : 41,
  "dcmp:frozenModelNames" : []
}
```

Freeze models – ask erocci by a POST (done in curl)

```
$ curl -v -H 'content-type: application/json' -X POST http://localhost:8080/collections/project/ -d @occiware_geo_0_published.json
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /collections/project/ HTTP/1.1
> User-Agent: curl/7.35.0
> Host: localhost:8080
> Accept: */*
> content-type: application/json
> Content-Length: 135
>
* upload completely sent off: 135 out of 135 bytes
< HTTP/1.1 201 Created
< connection: keep-alive
< date: Wed, 23 Mar 2016 12:01:38 GMT
< content-length: 0
* Server erocci OCCI/1.1 is not blacklisted
< server: erocci OCCI/1.1
< content-type: text/plain
< vary: accept
< location: http://localhost:8080/collections/project/2229d339-1889-3da8-9115-5ae853e77c5
```



Freeze models – erocci DBus to Java bridge calls Linked Data server REST API

BackendDBusService [Java Application] /usr/lib/jvm/jdk1.8.0_45/bin/java (23 mars 2016 13:02:05)

ID: 2

Address: http://localhost:8180/dc/type/dcmp:Project_0/geo_0

Http-Method: PUT

Content-Type: application/json

Headers: {Content-Type=[application/json], Accept=[application/json], X-Datacore-Project=[oas

Payload: {"@id":"http://data.ozwillo.com/dc/type/dcmp:Project_0/geo_0","o:version":38,"@type"

mars 23, 2016 1:07:59 PM org.apache.cxf.interceptor.LoggingInInterceptor

INFOS: Inbound Message

ID: 2

Response-Code: 200

Encoding: ISO-8859-1

Content-Type: application/json

Headers: {content-type=[application/json], Date=[Wed, 23 Mar 2016 12:07:59 GMT, Wed, 23 Mar 2

Payload: {"@id":"http://data.ozwillo.com/dc/type/dcmp:Project_0/geo_0","o:version":39,"@type"



At the end, models are frozen

As shown in Linked Data server backoffice (API Playground) :

```
localhost:8180/dc-ui/index.html | Rechercher

Datacore (http://localhost:8180):

/dc/type/dcmp:Project_0/geo_0 GET l dc

{
  "@id": "http://data.ozwillo.com/dc/type/dcmp:Project_0/geo_0",
  "o:version" : 42,
  "dcmp:frozenModelNames" : [
    *
  ],
}
```

What now in OCCIware ?

- **In the works :**

- Studio : more generators & connectors (ex. get current runtime state and update the diagram accordingly), backend generation, integrating simulator, decision-making tool...
- Runtime : complete erocci-dbus-java bridge, use it to integrate all backends behind erocci
- Deployment, monitoring, OCCI administration console
- use case platforms development / deployment / setup...
- and especially extending this Linked Data as a Service implementation to support all scenarii, including IoT ones !

Any questions ?

Thanks for your attention !

Contact : <http://www.occiware.org> - marc.dutoo at openwide.fr

Source : <https://github.com/occiware>

Partners :



Sponsors :

DGE (PIA) & System@tic, SCS, Images & Réseaux, PICOM, Minalogic clusters