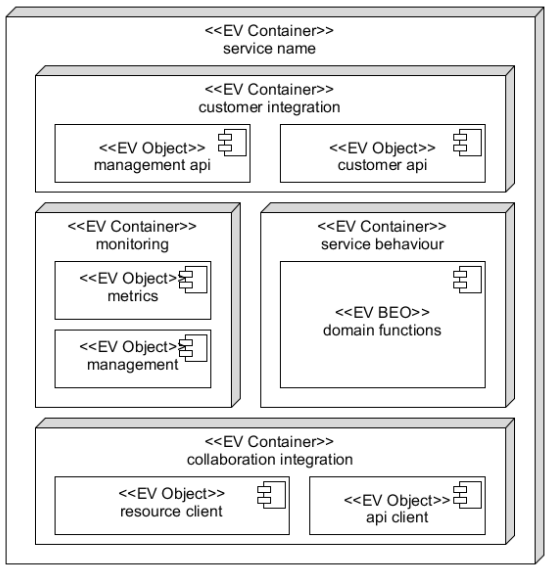
Container Structure (Draft)

The first element to describe in the engineering viewpoint is the structure of the **Container Objects** that help structuring the systems supporting the RI processes.

An **Container Objects** is an structural engineering object. **Container Objects** help structuring the systems supporting the RI processes. There are four possible types of container objects: node, nucleus, capsule and cluster [01].

The ENVRI RM engineering viewpoint recommendation defines the distribution of objects as services. In this configuration, EV objects are distributed in four main containers which group customer integration interface objects, collaboration integration interface objects, management components and domain function components.

- **Customer Integration**: provides interface components (APIs) which allow the use of the functionality by different service clients. This can include exposing management functionalities so that the service can be controlled remotely.
- **Downstream Integration (Collaboration Integration)**: provided to allow the service to call other services and back end components. This includes calling persistence layer components, such as storage manager components.
- **Service Management**: provided for management and monitoring of the service. This allows controlling the service (start, stop, pause, restart) as well as providing data for troubleshooting, logging, and provenance.
- **Service Behaviour**: provided to implement service behaviour. This is where the main functionalities of the service are located.



Recommended container structure for functionalities exposed as services.

The specific type of each container is left open to allow greater flexibility in distribution at implementation time.

The proposed architectural container structure for the EV is based on the Microservice Architecture [02, 03]. However, the EV does not propose a full migration to Microservices architectures but a gradual approach in which functionalities are decoupled and provided as services in a piecemeal fashion. For instance, functionalities which are always dependent on external services such as data identification and citation, or common functionalities which can be shared among various RIs, such as processing, cataloguing, provenance or or AAAI.

[01] See Linington et.al. p. 94-95 [**37**]

[02] Sam Newman. (2012). Building Microservices, Designing Fine Grained Systems. O'Reilly, ISBN-10: 1491950358

[03] Neal Ford Building Microservice Architectures. ThoughtWorks. Voxxed days Vienna 2016. Video https://www.youtube.com/watch?v=pjN7CaGPFB4, slides: http://nealford.com/downloads/Building_Microservice_Architectures_Neal_Ford.pdf