Exercise1. Modeling of the system infrastructure and software deployment.

The objective of this exercise is to measure the expressiveness of the DSL for modeling the infrastructure of an IoT system and the deployment of containerized-based applications. As part of this exercise, you will use the DSL to model part of an IoT system and answer a questionnaire at the end.

Figure 1 shows the infrastructure of an IoT system (excluding the device layer). This system is composed by edge, fog, and cloud nodes distributed in four regions (Region A, B, C, and D). The specifications of the nodes are summarized in Table 2. Each node host one or two software containers described in Table 1.

Using the DSL, model the following concepts.

1. Use Section 3 of the model template to specify the list of applications. You can see Table 1 to get the specifications for each app.
2. Use Section 4 of the model template to specify the list of nodes and their software containers.

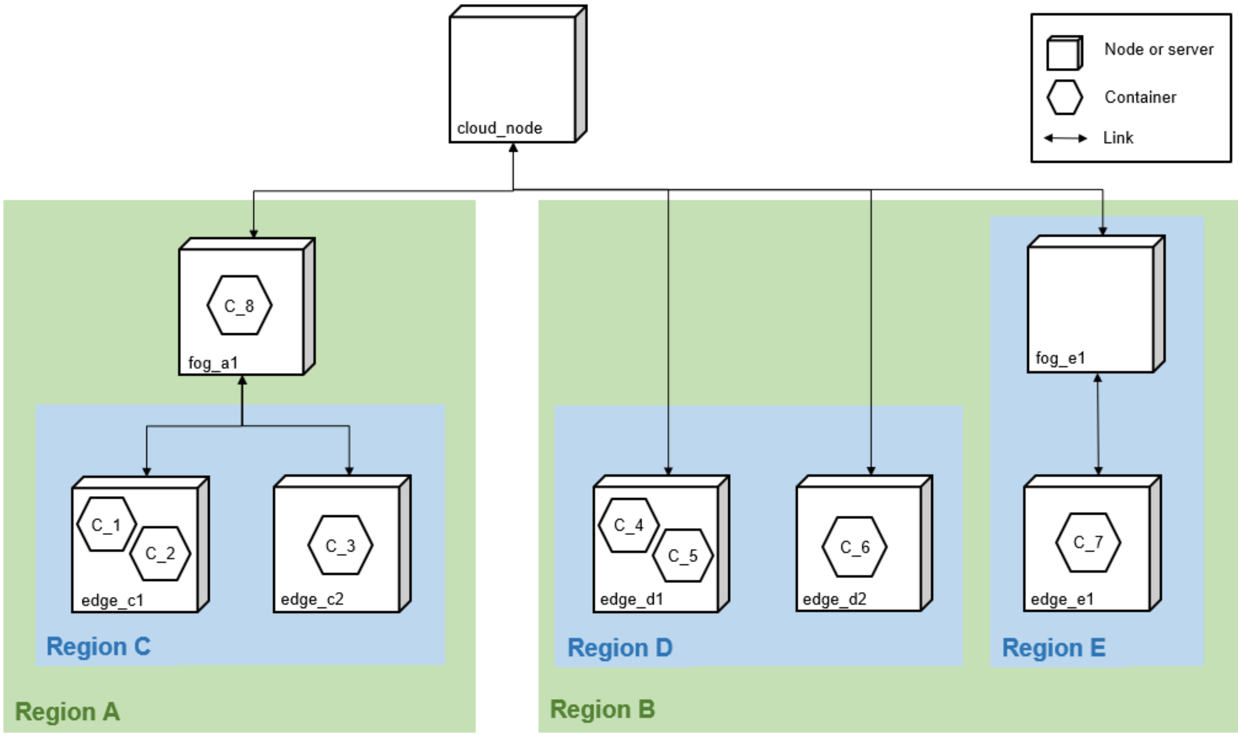


Fig 1. System infrastructure

Application	Application description	Container	Container requirements
App1	Memory required: 500 MB CPU required: 500 mCore Port: 8000 NodePort: 30021 Repository: iot-system/app1	C_1	Memory limit: no limit CPU limit: no limit
		C_3	
		C_4	
		C_6	
App2	Memory required: 700 MB CPU required: 700 mCore Port: 8081 NodePort: 30022 Repository: iot-system/app2	C_2	
		C_5	
		C_7	
App3	Memory required: 2000 MB CPU required: 1000 mCore Port: 8090 NodePort: 30023 Repository: iot-system/app3	C_8	Memory limit: 3000 MB CPU limit: 1500 MB

Note: containers do not have assigned volumes, so you must leave this field empty (< . . . >).

Table 1. Containerized applications description

Cloud node	Fog node	Edge node
Memory: 16000 MB Storage: 200000 MB CPU cores: 4000 mCore Operating system: Ubuntu Processor: x64	Memory: 4000 MB Storage: 50000 MB CPU cores: 2000 mCore Operating system: Ubuntu Processor: x64	Memory: 2000 MB Storage: 16000 MB CPU cores: 1000 mCore Operating system: Debian Processor: ARM

Table 2. Node’s specifications

Exercise 2. Adaptation rules modeling

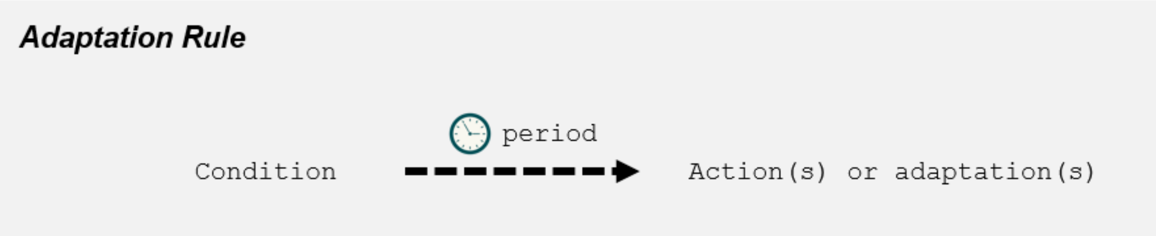
> *The objective of this exercise is to measure the expressiveness of the DSL for modeling the adaptation rules. As part of this exercise, you will use the DSL to model adaptation rules for the system described in the Exercise 1. Finally, you will answer a questionnaire.*

Based on the IoT system of Exercise 1, define the following adaptation rules using the DSL (Section 7 in the model template).

#	Condition	Period	Actions
1	*Unavailability* of **C_6**	20 seg	*Redeployment* of **C_6**
2	*CPU consumption* of **edge_c1** > 80 %	60 seg	*Offloading* of **C_2** to **fog_a1**
3	*CPU consumption* of **edge_d1** > 80 % or *RAM consumption* **edge_d1** > 70 %	2 min	*Offloading* **C_5** to **edge_d2** or *Scaling* **App2** (1 instance) in any node of ***Region B***
4	*Network throughput in* of **edge_e1** > 500 Mbps	50 seg	*Scaling* **App3** (2 instances) in **fog_e1** or in any node of ***Region B***
5	*CPU consumption* of any edge node in the ***Region C*** > 90 % or RAM consumption of any node in the ***Region C*** > 80 %	3 min	*Offloading* **C_8** to **cloud_node** and *Offloading* **C_2** to **fog_a1** and *Scaling* **App3** (2 instances) in any node of ***Region A***

Support Information

An adaptation rule is composed mainly of a condition, a period, and a list of actions. An adaptation rule indicates the list of actions taken when the condition is maintained during the period.



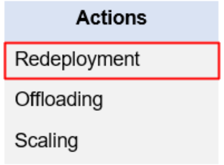
Conditions can be modeled using the following expressions:

Left expression	*Operator*	*Right expression*
Metric [*node / container*]	> < ==	*{numerical value}*
Metric [*region* –> *nodesLayer*]	>= <= !=	*true/false*

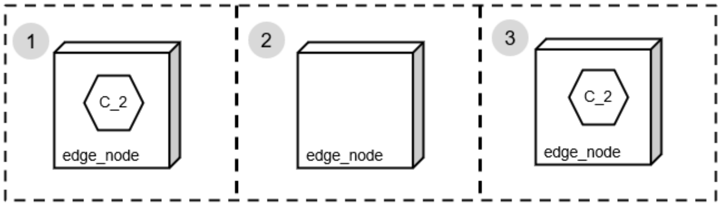
Examples of conditions:

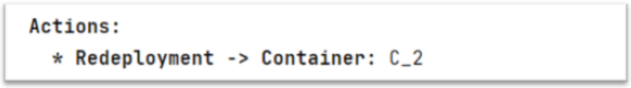
```
                        ( CPU [edge_node_01] )    >    ( 80% )
                 ( availability[container_01] )   ==    true
( net_throughput_in[edificioA –> edge_nodes] )    >=   ( 500 Mbps )
 ( net_throughput_in[edificioA –> all_nodes] )    >=   ( 500 Mbps )
```

For this exercise, you will use three kinds of adaptations or actions (redeployment, scaling, and offloadig) summarised below.

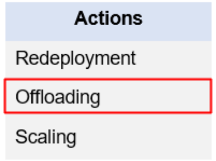


Actions
Redeployment
Offloading
Scaling

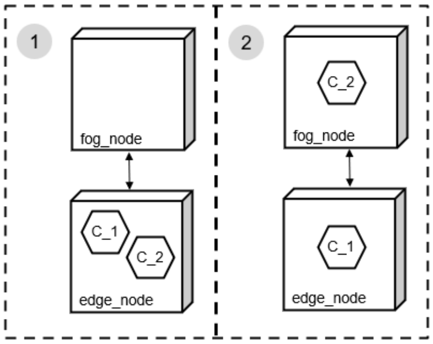


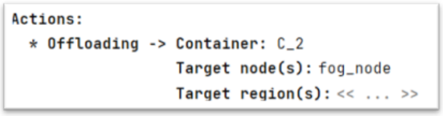


```
Actions:
  * Redeployment -> Container: C_2
```



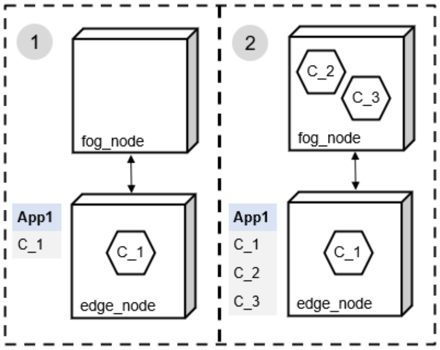
Actions
Redeployment
Offloading
Scaling

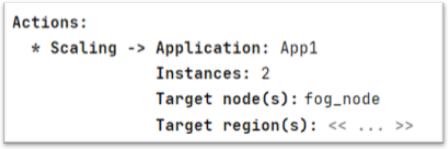




```
Actions:
  * Offloading -> Container: C_2
                  Target node(s): fog_node
                  Target region(s): << ... >>
```

Actions
Redeployment
Offloading
Scaling





```
Actions:
  * Scaling -> Application: App1
               Instances: 2
               Target node(s): fog_node
               Target region(s): << ... >>
```