QoS-aware Deployment of Service Compositions in 5G-empowered Edge-Cloud Continuum

Marco Anisetti, Filippo Berto, Ruslan Bondaruc
Department of Computer Science
Università degli Studi di Milano, Milan, Italy
{firstname.lastname}@unimi.it

***Abstract*—Nowadays, modern service compositions are increasingly adopted in critical scenarios where advanced Quality of Services (QoS) such as low latency, security, and privacy are fundamental. The landing platforms for the deployment of such compositions are progressively becoming capable to offer capabilities that support such advanced QoS requests (e.g., low latency via 5G network slice) spanning the Edge-Cloud Continuum. Actual deployment solutions focus mainly on resource allocation (i.e., CPU, memory, and storage), falling short of addressing advanced QoS and unleashing the true potential of the Edge-Cloud Continuum. In this paper, we present an automatic QoS-aware deployment solution for composed services in the Edge-Cloud Continuum. It compares QoS requests on the service composition with the capabilities of a given continuum in order to find, generate and execute suitable deployment recipes. Our preliminary experimental evaluation demonstrates the feasibility of our solution in a realistic scenario.**

***Index Terms*—Service Composition; Service Deployment; Non-Functional properties; 5G MEC; Edge-Cloud Continuum**

I. Introduction

Cloud-native technologies are drastically improving the way infrastructure resources are allocated and applications are deployed. For developers, the ability to encapsulate applications in containers or virtual machines simplifies the development process, making it independent of and transparent to deployment. Containerization allows also to build applications following the distributed paradigm, where components, mainly services, are chained into complex compositions.

The advent of Edge Computing has changed the way a service composition can be deployed, allowing for instance to better support real-time applications via low latency [1] and in general a better placement of services to fulfill specific QoS requests [2], [3]. Despite being physically separated from the Cloud, the Edge heavily depends on it to execute resource-hungry tasks, since computational power available on Edge nodes decreases the farthest they are from the Cloud. This dependency has been emphasized in recent times, when the momentum gained by machine learning (ML) pointed out the computational limitations of Edge Computing and consequently stimulated the development of distributed approaches relying on both Cloud and Edge, such as federated learning (FL) [4].

In such context, a new computational paradigm is taking hold, merging both Cloud and Edge in a Continuum. Edge-Cloud Continuum combines the strengths from both platforms in terms of scalability, flexibility, and mobility, to name but a few, and delivers a wide infrastructure where applications can be seamlessly deployed across the entire network [5]. The aim of the Continuum is to decouple the management of heterogeneous resources and the deployment configuration from the execution of the specific application. This decoupling can be achieved by abstracting the deployment infrastructure, allowing the developer to simply define applications in terms of functional capabilities (i.e., a workflow of services) and QoS properties to hold (e.g., confidentiality, availability, latency, etc.). In our approach we assume that most of such non-functional properties can be guaranteed thanks to suitable deployment configurations, without the need of modifying the application logic. For instance, considering an application on the continuum made of a pipeline of services focused on collecting sensible data to build a specific model. Low latency and privacy properties can be granted for the data gathering services via on premises or edge deployment, while high computational power can be guaranteed for the data modeling services via cloud deployment. We note that in a Cloud-only or Edge-only scenario, we may not benefit from all the properties without requiring modification at application logic, whereas in the Continuum, the same requirements may be fulfilled by specific deployment recipes.

Notwithstanding the potentialities offered by the Edge-Cloud Continuum, the research into how to fully exploit its potential is still in its infancy. Some solutions have been proposed in the field of serverless computation, where stateless applications are executed in modern Function-as-a-Service (FaaS) platforms [6], [7], [8]. In FaaS, frameworks are developed to select the best platform according to some metrics, such as cost. However, stateless applications focus only on resource allocation (i.e., CPU, memory and bandwidth) and represent only a portion of a much wider and complex range of Edge-Cloud Continuum applications requiring much more complex properties to hold. For instance, the metrics defined for the selection of the optimal FaaS platform cannot be used to cope with latency, confidentiality and authentication [9], [10], [6], [7]. In short, we still lack a non-functional property-aware approach to deploy applications in the Edge-Cloud Continuum, since existing solutions work mainly for serverless frameworks and fail short to handle pipelines of services and complex properties.

Our paper aims to address the above gap by defining a new methodology for service deployment in the Edge-Cloud Con-

tinuum. We extended the graph-based representation of service composition in [11] to model also the Continuum deployment environment peculiarities. We then build a matching function capable to map each service in the composition to a facility of the Continuum, so that all the required properties can be fulfilled. This mapping is finally used to generate and deliver deployment recipes specific for each Continuum facility.

The contribution of this paper is fourfold: i) a new notion of Edge-Cloud Continuum involving both 5G Telco Edge node and on-premises Edge node (Section II), ii) a novel methodology for QoS-aware deployment of composed services in the Edge-Cloud Continuum (Section III), iii) a suitable Architecture implementing the methodology and generating executable recipes for the deployment, and iv) preliminary experimental evaluation showing the feasibility (Section IV).

II. Scenario, Requirements and Architecture

Our reference scenario considers i) a client that wants to deploy its workflow of services s_i specifying QoS requirements and constraints to be satisfied on the Edge-Cloud Continuum; ii) Cloud Service Providers (CSPs) offering deployment facilities f_i for third-party services on the Edge-Cloud Continuum; iii) facilities offering specific capabilities c_i to satisfy QoS requirements and constraints. In this paper we consider an advanced Edge-Cloud Continuum where Edge nodes can be: i) telco nodes (i.e., 5G Multi-access Edge Computing - MEC) based on an agreement between the CSP and a given telco operator offering their core network capabilities to be part of the CSPs Continuum (e.g., AWS Wavelength); ii) on-premises nodes based on the deployment facilities on the client's premises enabled by the CSP for services deployment (e.g., using AWS Greengrass). When the client asks to deploy a given workflow of services, the CSP matches the service workflow, QoS and constraints with the capabilities and peculiarities of its Edge-Cloud Continuum deployment facilities associated to the specific client in order to find all the feasible deployment configurations. Among them the CSP selects the configuration to be deployed according to internal policies, such as considering Cloud as the preferred service deployment due to lower operating costs. For instance, in case of two candidate configurations, one using 5G and Cloud facilities and another using on-premises and Cloud facilities, the latter would be selected. The CSP then generates the deployment recipe for the selected configuration and executes the deployment on the continuum.

Example II.1 shows the scenario used in the rest of the paper to present our methodology.

Example II.1 (The Scenario)**.** Let us assume that a client wants to deploy a machine learning workflow made of 4 sequential services: s_1 gathering data, s_2 normalizing data, s_3 generating a model out of the collected data (e.g., training a decision tree), and s_4 saving the model for further usage (e.g., for prediction). Let us assume that the client requires that data accessed by s_1 should be protected for confidentiality and the model generated by s_4 should be protected from tampering by controlling its integrity. Let us assume that the client expresses a constraint on the communication links between s_1 and s_2 and between s_2 and s_3, which must carry large volumes of row data, requiring a bandwidth of at least 200 Mbit/s. Let us also assume the client has a contract with the CSP for Continuum facilities including on-premises machine f_1, 5G Edge node f_2 in the proximity of its premises, and Cloud node f_3. The CSP ensures some capabilities (c) on the facilities and the network links connecting them. In particular, f_1 ensures confidentiality at rest via isolation of the storage from the direct control of CSP (c_1), and f_3 provides a service (s_I) ensuring data integrity at rest (c_2). In addition, all the internal links, that is the links between two services deployed on the same facility, have infinite bandwidth (c_3), while the 5G link between f_1 and f_2 provide at least a 500 Mbit/s bandwidth (c_4).

A. Edge-continuum deployment Requirements

In order to support the Scenario in Section II, the CSP should be empowered with an advanced deployment architecture addressing the following requirements:

- **(R1) Continuum-readiness**: it should seamlessly deploy services on all the different Continuum premises.
- **(R2) Property-driven**: it should be driven by QoS properties and constrains expressed by the client.
- **(R3) Technology agnostic**: it should be capable to handle heterogeneous deployment facilities regardless the underlying virtualization technology.
- **(R4) Comprehensive model**: it should provide a way as general as possible to represent pipelines and facilities, without limiting the choice in topology and data flow.
- **(R5) Interoperability**: it should be able to interact with CSP facilities through software hooks.
- **(R6) Context adaptability**: it should perform deployment life-cycle management by re-deploying services when changes in the environment occur.

To the best of our knowledge, no deployment architecture is currently capable to fully address the above requirements offering a comprehensive and adaptable approach to permanently deploy services in the Continuum taking into account client-defined advanced properties and constrains. Few solutions exist for application deployment, which are compared in Table 1 with respect to the above requirements (full and partial support of a requirement is denoted with ✓and $\sim$ respectively). Some considered works ([12], [13], [14], [9]) address the Edge-Cloud Continuum scenario, but only the architecture presented in [14] can seamlessly deploy applications along the Continuum since it is fully independent of the specific technology and CSP involved. QoS requirements and constraints are taken in account, even partially, by most of the works ([13], [15], [14], [6], [16], [10]), but only [13], [15] provide a comprehensive way to model both applications and environment. Finally, there is no solution that performs a life-cycle management of the deployment, adapting to changes in context. In short, all solutions have drawbacks, whether they be the limited application scenario, the dependence on specific technologies, or the inadequate composition model.

Table 1: Related work comparison.

Author	Ref.	R1	R2	R3	R4	R5	R6
K. Fu et al.	[12]	✓			✓		∼
A. Orive et al.	[13]	✓	∼		✓	✓	
A. Brogi et al.	[15]		∼	✓	✓		
V. Casola et al.	[14]	✓	✓	✓		✓	
S. Nastic et al.	[9]	✓		✓			
N. Akhtar et al.	[6]		∼	✓		✓	∼
A. Das et al.	[8]			✓	✓	✓	
M. Anisetti et al.	[16]		✓	✓			
J. Quenum et al.	[10]		✓	✓		✓	
Our Work		✓	✓	✓	✓	✓	✓

Our architecture, instead, addresses all the aforementioned requirements, providing a QoS-driven deployment system for the Continuum.

B. Deployment Architecture

To support the scenario in Section II and ensure the requirements in Section II-A, we propose a deployment architecture capable of guaranteeing a seamless QoS and constraints preserving execution of a given service workflow in the Continuum.

Figure 1 schematizes our system architecture made of i) a *Deployment Engine* service that targets the deployment facilities through *Deployment Agents*, and ii) a set of *Deployment Facilities* of different nature, including Cloud, Telco-Edge and on-premises nodes (R1). The client interfaces with the deployment engine through *Deployment API* providing the service workflow to be deployed and the QoS/constraints specification in a machine-readable format (R2) and (R4). The *Deployer Solver* chooses the most appropriate deployment configuration interacting with the *Deployment Controller* which is in charge of i) interrogating Deployment Facilities on their capabilities and ii) delivering the deployment recipes, using the *Deployment Agents*.

The Deployment Agents are responsible to build the deployment continuum across the facilities by driving the service deployment (R3). In order to support QoS and constraint-aware integration, the CSP exposes to the Deployment Agents suitable hooks to relevant resource (e.g., resource manager for deployment) or services (e.g., services offering security features) constituting their capabilities (R5). For instance, the CSP can offer a hook to access non-functional certificates (i.e., using certification scheme in [17], [18]) proving some capabilities or to invoke authentication services to support authentication requirements.

We note that in case of necessity (e.g., changes in the QoS or budget constraints) the given workflow can be re-deployed via re-executing the deployment matching with modified QoS and constraints (R6). The re-deployment can be also used to handle service migrations and in general changes occurring post-deployment.

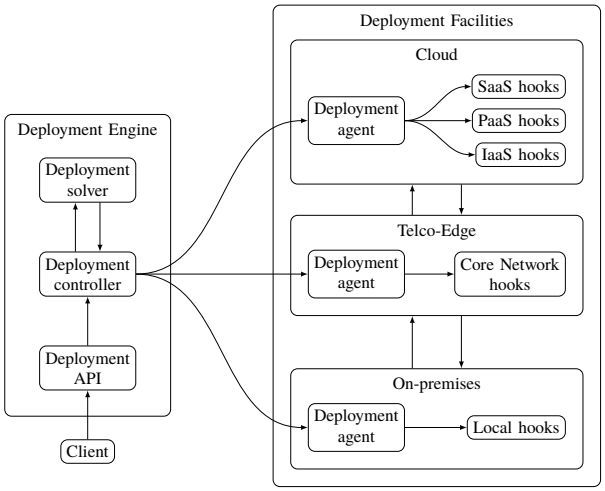

Fig. 1: Deployment Architecture for Edge-Cloud Continuum.

In the following we describe the peculiarities of the Continuum *Deployment Facilities* in terms of architecture and capabilities.

C. Cloud

Cloud solutions offer to their users managed services and infrastructure with scalable amount of resources. This allows users to deploy and integrate their services and products, relieving them from most management tasks. Generally, the control plane of the system is handled by a cloud provider, managing the services life-cycle and ensuring their availability. The most common type of deployment used in this context are based on containerized services or on virtualized hosts.

The Cloud is the most flexible and powerful facility in the Continuum in terms of offered capabilities. It supports non-functional QoS via PaaS or IaaS services enabling them for the deployed services. It also partially supports constraints, for instance via resource scalability, network latency, high bandwidth and connectivity supported by virtualized network and local and distributed data centers. However, this support is normally very limited due to the public internet connectivity and the shared environment used in standard Cloud configurations. The Cloud currently suffers from the lack of transparency impacting non-functional properties such as privacy [19].

To support our deployment architecture the Cloud facility has to provide hooks for i) resource management offering deployment and configuration of containerized and/or virtualized services; ii) configurations of services relevant to support given non-functional properties; and iii) workflow-level networking management.

D. Telco-Edge

Telco-Edge is an innovative Edge scenario enabled by 5G via integrating mobile networking capabilities. The state-of-the-art solution for automated service management in such systems is based on MEC [20], [21], [22], [23]. MEC allows integration of container and VM based services with the 5G

core network, mutually exposing their functionality. Differently from traditional Cloud facilities, i) the edge nodes are connected to 5G radio antennas, allowing fast communication to and from mobile devices; ii) the service provisioning to mobile devices is backed by unique device and user identification through IMEI and SIM identifiers, supporting strong authentication; iii) the telco edge nodes can be geographically closer to their users, enabling new notion of data privacy by proximity, similarly to the notion of privacy on-premises; and iv) the 5G standard allows users to allocate virtual network slices, ensuring bandwidth availability and network performance levels and latency.

In addition, by adopting the Telco-Edge facilities i) the bandwidth between the edge node and other network nodes is allocable; ii) the in-motion data can be contained within the boundary of the telco network; and iii) additional services (i.e., identification, network monitoring, authentication) for the users connected through mobile network are available.

To support our deployment solution, the Telco-Edge facility have to offer hooks on resource management and service configuration similarly to the Cloud but also additional peculiar hooks for i) network-level user management and authentication, and ii) network resource allocation (5G network slices) for bandwidth and latency management via MEC.

E. On-premises

With on-premises we consider deployment facilities that are fully under the control of the owner, but are equipped with CSP services to make them part of the Continuum. They can be realized via VMs/containers or physical machines. Such facilities have normally stronger resource limitations compared to Cloud or Edge environments, lacking scalability and elasticity or not providing specific hardware. On the contrary, they i) have strong properties of high confidentiality and privacy, for instance ensuring that data cannot physically leave the organization; ii) have the lowest latency to the devices within the organization; and iii) leave the owners full control of the execution environment.

In order to be part of the Continuum, on premises facilities should i) allow our Deployment Agents to be executed in a supported deployment environment; ii) access to resources to arrange the service deployment; and iii) offer hooks to handle resources for local deployment of services, connectivity to the continuum and access to the relevant local services if needed. Normally, on premises hooks have very restricted access to local services, data and deployment resources making their use in the continuum very challenging.

III. Methodology

Figure 2 shows our methodology for the Edge-Cloud Continuum scenario in Section II based on our architecture in Figure 1. The client defines the service composition workflow to be deployed and annotates it with QoS requirements and constraints forming an *Annotated Service Composition Template*. The service provider annotates its facilities with the relevant properties, indicates hooks reachable by Deployment Agents, and produces the *Annotated Deployment Facilities Graph*. The client submits the request for deployment by submitting the Service Composition Template via Deployment API to our Deployment Engine. The Service Composition Template and the Annotated Deployment Facilities Graph triggers the *Deployment Matching* process executed by our Deployer Solver in order to generate the *Deployment Recipes* to be used to deploy the given workflow of services on the Edge-Cloud Continuum.

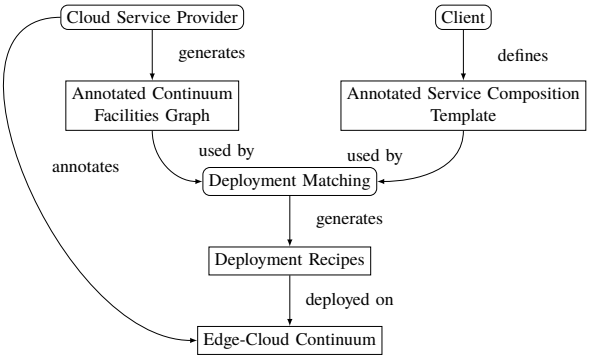

Fig. 2: Our Methodology.

A. Annotated Service Composition Template

A Service Composition Template is an abstract representation of the workflow of services that a client wants to deploy. It describes the services, their execution parameters and their interconnectivity configuration.

Definition III.1 (Service Composition Template)**.** A Service Composition Template is a directed graph $T = (S, E)$ where $s_i \in S$ are services constituting the graph vertexes, and $e_i \in E$ are graph edges modeling the interaction between two services on both control and data plane. We note that, since $T = (S, E)$ is a directed graph, each edge represents a one-way interaction, while a two-way communication requires a cycle made by two edges.

The Service Composition Template can be annotated with specific non-functional requirements $r_i \in R$ and constraints $k_i \in K$ on resources. Following the notation in [24], a requirement r_i is a pair $(\hat{r}, Attr)$, where $r_i.\hat{r}$ is an abstract requirement from a shared vocabulary of properties (e.g., confidentiality, integrity) and $r_i.Attr$ is a set of attributes specifying the low-level characteristics that should be provided. The attribute values induce a hierarchy H_R of requirements $(R, \preceq_R)$, where R is the set of requirements and $\preceq_R$ is the partial order. Similarly, a constraint $k_i \in K$ is a tuple $(\hat{k}, Val, Op)$, where $k_i.\hat{k}$ is a resource (e.g., bandwidth, on-premises), $k_i.Val$ is the desired value and $k_i.Op$ is the operation on that value. The resource determines what type of value can be specified, such as integers, booleans, lists, and what operations can be applied (e.g., $=, <, \geq, \in$).

Definition III.2 (Annotated Service Composition Template)**.** Let $T = (S, E)$ be a Service Composition Template. The

Annotated Service Composition Template $T^{R,K}$ is generated annotating vertexes and edges in the template $T = (S, E)$ with non-functional requirements $r_i \in R$ and constraints $k_i \in K$.

For instance, a security requirement r_i = (*Confidentiality*, *AES256*) can be associated to an edge (i.e., communication channel) e_i, denoted as $e_i^{r_i}$, while a constraint k_j = (*On-premises*, `true`, $=$) can be associated to a vertex (i.e., a service of the workflow) s_j denoted as $s_j^{k_j}$ indicating that the deployment must be performed on an on-premises facility.

Example III.1 (Annotated Service Composition Template)**.** Considering Example II.1, we can represent the client service workflow as the annotated template $T^{R,K}$ with the set of services $S = [s_1^{r_1}, s_2, s_3, s_4^{r_2}]$ and the set of edges $E = [e_1^{k_1}, e_2^{k_1}, e_3]$. The expressed requirements are r_1 = (*Confidentiality*, *Isolation*) and r_2 = (*Integrity*, *Rest*), while the posed constraint is k_1 = (*Bandwidth*, 200, $\geq$).

B. Annotated Continuum Facilities Graph

The *Cloud Service Provider* (CSP) models its Continuum facilities for a given client as a Continuum Facilities Graph.

Definition III.3 (Continuum Facilities Graph)**.** The Continuum Facilities Graph is a directed graph $G = (F, L)$ made of a vertex $f_i \in F$ for each facility provided by the CSP. L is the set of edges (here called links) l_i connecting the vertices of the graph. Two vertices f_i and f_j can be connected by a link l_i if facility f_j is reachable from facility f_i either through the open Internet or a dedicated private channel.

Note that the Continuum Facilities Graph is directed and therefore each link models a one-way connection. We can however express both two-way and intra-node communication capabilities by defining cycles. In particular, intra-node communication is represented as a cycle of length 1, i.e., a loop.

The Continuum Facilities Graph generated is then annotated with non-functional capabilities $c_i \in C$. Capabilities represent a mechanism to support both non-functional properties and constraints such as confidentiality, latency, performance to name but a few. A capability is defined as a tuple $(\hat{c}, Spec, Op, Impl)$, where $c_i.\hat{c}$ is either a property or a resource, $c_i.Spec$ is an attribute if $c_i.\hat{c}$ is a property, a value otherwise, $c_i.Op$ is an operation on $c_i.Spec$ (if it is an attribute, Val is always a $=$), and $c_i.Impl$ is a set, even empty, of key-value pairs describing how the facility implements the capability. The annotated data life-cycle is managed by the service providers, possibly using certification-based solutions. The selection and implementation of the associated methodology is out of the scope of this paper.

Definition III.4 (Annotated Continuum Facilities Graph)**.** Let $G = (F, L)$ be a Continuum Facilities Graph. The Annotated Continuum Facilities Graph G^C is generated annotating vertexes and edges in the Continuum Facilities Graph $G = (F, L)$ with the capabilities $c_i \in C$ (e.g., latency, bandwidth, resources).

For instance, a security capability c_i = (*Channel_encryption*, *AES256*, $=$) can be associated to a link l_i denoted as $l_i^{c_i}$, while a capability c_j = (*Edge*, `true`, $=$) can be associated to a vertex (i.e., a facility of the SP) f_j denoted as $f_j^{r_j}$.

Together, Annotated Service Composition Template and Annotated Continuum Facilities Graph provide a general framework to describe most kind of pipeline topologies in the Continuum, addressing (R4).

Example III.2 (Annotated Continuum Facilities Graph)**.** Considering Example II.1, we can represent the CSP facilities as the annotated graph G^C with the set of facilities $F = [f_1^{c_1}, f_2, f_3^{c_2}]$ and the set of links $L = [l_1^{c_3}, l_2^{c_4}, l_3^{c_3}, l_4, l_5, l_6, l_7^{c_3}]$. The provided capabilities are c_1 = (*Confidentiality*, *Isolation*, $=$, []), c_2 = (*Integrity*, *Rest*, $=$, [*service*: *sI*; *mode*: *interception*]), c_3 = (*Bandwidth*, $+\infty$, $=$, []) and c_4 = (*Bandwidth*, 500, $\geq$, []).

C. Deployment matching

The deployment matching process searches for the most suitable solution for the QoS-aware deployment of a given service workflow on the Continuum. It takes as input the Annotated Continuum Facilities Graph G^C and the Annotated Service Composition Template $T^{R,K}$, generates a set of suitable deployment solutions M and among them finds the one $\hat{M}$ that better satisfies a given CSP policy (e.g., the lowest operational cost).

The set of suitable deployment solutions M is defined on $(S \in T^{R,K}) \times (F \in G^C)$ so that: i) for every edge $e_i \in E$ between any two vertices s_i and $s_j \in S$ there is a link $l_i \in L$ between the matching vertices f_i and $f_j \in F$; ii) for every pair (s_i, f_i) in M, the capabilities c_i of f_i satisfy the requirements r_i and the constraints k_i of s_i; iii) for every edge e_i between any two vertices s_i and $s_j \in S$, the capabilities c_i of l_i between the matching vertices f_i and $f_j \in F$ satisfy the requirements r_i and the constraints k_i of e_i. If the set of suitable deployment solutions M is empty, it means that the deployment of the service workflow cannot take place on the given continuum, given the QoS requirements and constraints. If the set of suitable deployment solutions M is not empty, the deployment matching orders them according to the CSP policy and selects the first one in the order.

Given the set of deployment solutions M, if the CSP policy refers to operational cost reduction only, a solution like the one in [16] can be adopted. We will investigate the impact of more articulated CSP policies as well as the adoption of an optimization approach for finding the suitable deployment solution and contemporaneously addressing such CSP policy in our future works.

Algorithm 1 shows the pseudocode of our matching function. The algorithm i) iterates over all the permutations of services altering the elements starting from the beginning of the list; ii) for each permutation it iterates on its partitions starting from the beginning of the list, thus leaving the largest partitions containing the most preferred facilities; iii) for each permutation it checks whether all requirements are met, if that

Algorithm 1 Pseudocode of our deployment matching exhaustive algorithm.

```
procedure MATCHING(S, F)
    matches = ∅
    ▷ Iterate services permutations
    for service_perm in perm(S) do
        ▷ Services partitioning in |F| facilities
        for part in partitions(service_perm, |F|) do
            ▷ Test if all requirements are met
            valid ← true
            for r_i in R do
                if r_i(S, F, part) == false then
                    valid ← false
                    break
                end if
            end for
            if valid then
                ▷ Match found
                matches = matches ∪ {part}
            end if
        end for
    end for
    return matches
end procedure
```

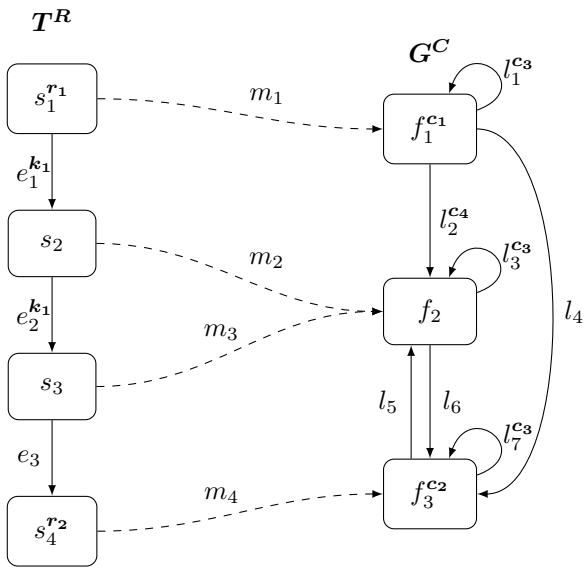

Fig. 3: Annotated Deployment Graphs including (a) Annotated Service Composition Template and (b) Annotated Continuum Facilities Graph, with the resulting matching.

is the case it adds it to the results partition otherwise it breaks to the inner for loop; iv) finally it returns the set of results.

Example III.3 (Deployment matching)**.** Let us consider the scenario in Example II.1, the Annotated Template $T^{R,K}$ and the Annotated Graph G^C, in Example III.1 and Example III.2 respectively. Let us now apply our matching function in Algorithm 1 to the services in $T^{R,K}$ and the facilities in G^C retrieving a set of possible matching M. Figure 3 shows one of the possible solution in $M = \{m_1, m_2, m_3, m_4\}$, where $m_1 = (s_1, f_1)$, $m_2 = (s_2, f_2)$, $m_3 = (s_3, f_2)$, and $m_4 = (s_4, f_3)$. Here, s_1 is matched with f_1 since it is the only facility ensuring confidentiality. Similarly, s_4 is matched with f_3 since it is the only facility providing a way to check integrity. Lastly, s_2 and s_3 are matched with f_2 to provide large bandwidth between the first three services.

D. Deployment Recipes

The purpose of the above deployment matching is to assign each service to a facility in such a way that all requirements (i.e., QoS and constraints) can be fulfilled and the CSP internal policy satisfied. However, a mere assignment is not always enough to enforce the desired properties. For instance, low latency can be achieved by simply assigning services to the physically closest facility, while, on the contrary, communication channel confidentiality requires configuring a component or service to handle message encryption and decryption. To address the above deployment challenges, our methodology enriches traditional deployment recipes with hooks metadata. This metadata is consumed by our Deployment Agents to enable relevant properties configuring or embedding facility's services in the service workflow to be deployed.

Recipes are generated for each service of the workflow in $\hat{M}$ and contain the operational instructions for the deployment of both the service and the supporting facility's components/services. In particular, a recipe consists of three parts: i) the deployment configuration of the service, including the image to be used and resources to be allocated; ii) a description of the support components/services to be integrated and their parameters (e.g., for an authentication component it includes the list of user credentials); iii) modality of integration (i.e., none, interception or wrapper).

Example III.4 (Deployment Recipe)**.** Considering the selected matching $\hat{M}$ in Example III.3, according to our methodology the relative deployment recipes are generated for all the services $s_i \in \hat{M}$. For simplicity let us focus on s_4 recipe only. Figure 4 shows an excerpt of the s_4 recipe provided to the deployment agent in f_3, along with the final deployment graph for $\hat{M}$. According to the recipe, the agent, in order to guarantee integrity, has to deploy an additional facility's service (*service:* s_I) that intercepts (*mode: interception*) incoming data from s_3, computes and adds to data a cryptographic checksum, and delivers it to s_4 for storing.

We note that details on how the recipes are generated out of the given selected matching $\hat{M}$ is out of the scope for this paper. We will further investigate this topic in our future works.

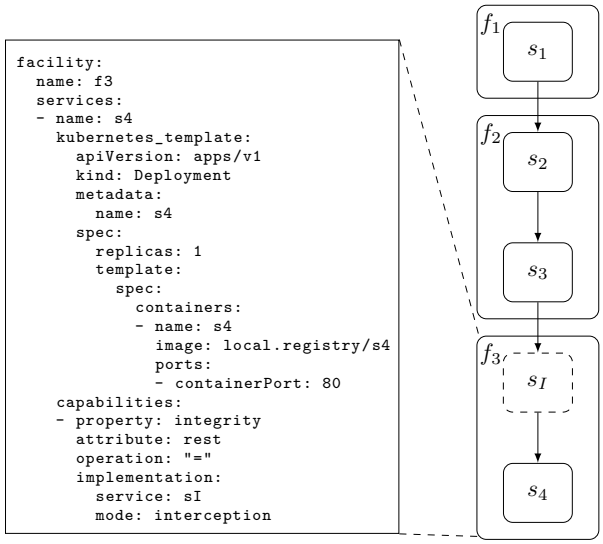

Fig. 4: Deployment Recipe for facility f_3.

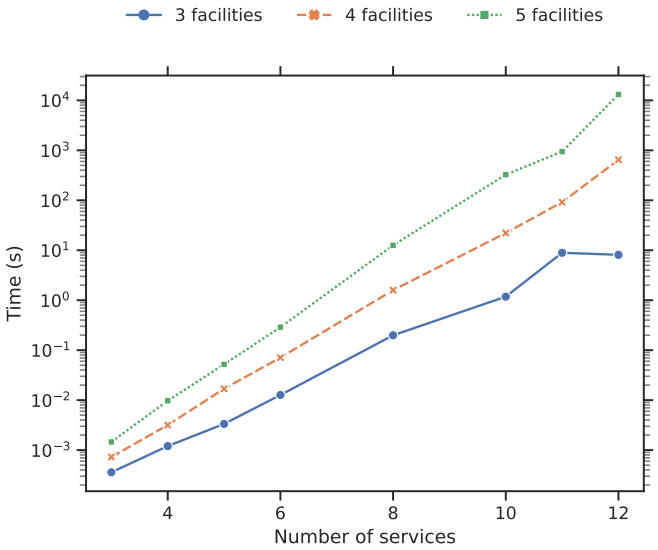

Fig. 5: Execution time with increasing number of services s_j in the workflow and considering 3, 4 and 5 facilities f_j.

IV. Experimental Evaluation

In this section we first describe our experimental settings realizing the scenario in Section II and then present a preliminary performance evaluation.

A. Experimental setup

We realized the Edge-Cloud Continuum of our scenario in Section II as follows. The software stack used in the Cloud facility was based on MicroK8s. It allowed to simulate cloud-hosted VPS nodes in a cluster formation supporting auto-scaling as we expect by a Cloud provider. The Telco-Edge facility was implemented based on Open Source MANO (OSM)[1], Free5GC[2] and MicroK8s[3], respectively implementing the MEC, a simulated 5G core network, and the resource manager. OSM was designed to integrate with the core network, exposing the services hosted in Kubernetes to the users and other nodes as Network Functions. The on-premises facility was based on MicroK8s, configured to handle limited resources. All the nodes were hosted in the same data center, and were equipped with 4 virtual cores clocked at 2.09GHz and 32 GB of RAM and running Linux 5.4.0 (Ubuntu 20.04 LTS). All the nodes of the continuum were empowered by our deployment agents communicating with our Deployment Engine using the same virtualized network used by the continuum node.

Our Deployment Engine was implemented using Python 3.10.9 and has been executed on a machine running Linux 5.15.102 (NixOS) equipped with an AMD 5900x and 32 GB of RAM.

B. Performance evaluation

The computational effort needed to deploy a given service workflow using our Deployment Engine is clearly dominated by the effort required by our matching function that can be estimated as $\mathcal{O}(|F|^{|S|})$ where $|F|$ and $|S|$ are the cardinality of Continuum facilities and services in the workflow respectively.

Figure 5 shows the execution time requested by the matching function varying the number of services $s_i \in S$ in the workflow and the number of facilities $f_j \in F$ of the continuum using logarithmic scale. The performances were computed on the average of 5 executions for each combination of services and facilities. For each execution we randomly generated facilities' capabilities and services' requirements. As expected the number of services $|S|$ dominates the exponential growth of execution time.

The matching algorithm terminates in less than one second when evaluating the cases of 6 services in 5 facilities and 8 services in 3 facilities. As the number of services increases, the time of execution grows rapidly: the execution time with 12 services and 3 facilities is 8.11s, which is the last experiment configuration to terminate under 10s. The cases 8 services in 5 facilities and 10 services in 4 facilities terminate respectively in 12.6s and 22.1s. With larger numbers of facilities and services the algorithm becomes too slow for practical use: the configuration 11 services in 4 facilities terminates in 91.7s. The last configuration shown in Figure 5 considers 12 services in 5 facilities and has terminated in 13143s.

V. Conclusions

In this paper we described a novel methodology for deploying composed services in Edge-Cloud Continuum focused on guaranteeing advanced QoS requirements. The main idea is to exploit the capabilities and the peculiarities of the different continuum landing platforms to ensure QoS at deployment time. Our deployment methodology is based on a machine-readable description of the service composition annotated with the QoS requirements and a meta-description of the

[1]https://osm.etsi.org/
[2]https://www.free5gc.org
[3]https://microk8s.io

capabilities of the landing platforms involved in the continuum. Based on these machine-readable descriptions, a set of feasible deployment solution are generated and the relative deployment recipes for the service composition are selected according to a given policy. We show the feasibility of our solution in our preliminary experimental evaluation. In our future works, we plan to investigate different deployment optimization solutions and consider a more complex experimental scenario involving migrations of service and data as well as changes at the network topology level.

Acknowledgement

This work is partly supported by the program "piano sostegno alla ricerca" funded by Università degli Studi di Milano and by the project MUSA - Multilayered Urban Sustainability Action - project, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strenghtening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037). The work was also partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NextGenerationEU. Filippo Berto acknowledges support from TIM S.p.A. through the Ph.D. scholarship. This project was also supported by a grant from Leonardo S.p.a.

References

[1] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[2] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, 2020.

[3] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani, "An assurance framework and process for hybrid systems," in *Communications in Computer and Information Science*, vol. 1484 CCIS, 2021, pp. 79–101.

[4] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, pp. 1–7, 2020.

[5] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The Internet of Things, Fog and Cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, 2018.

[6] N. Akhtar, A. Raza, V. Ishakian, and I. Matta, "COSE: Configuring Serverless Functions using Statistical Learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 129–138, iSSN: 2641-9874.

[7] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan, "A Serverless Real-Time Data Analytics Platform for Edge Computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

[8] A. Das, A. Leaf, C. A. Varela, and S. Patterson, "Skedulix: Hybrid Cloud Scheduling for Cost-Efficient Execution of Serverless Applications," in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 609–618, iSSN: 2159-6190.

[9] S. Nastic, P. Raith, A. Furutanpey, T. Pusztai, and S. Dustdar, "A serverless computing fabric for edge & cloud," in *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, 2022, pp. 1–12.

[10] J. G. Quenum and J. Josua, "Multi-cloud serverless function composition," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–10.

[11] M. Anisetti, F. Berto, and M. Banzi, "Orchestration of data-intensive pipeline in 5g-enabled edge continuum," in *Proc. of 2022 IEEE World Congress on Services, SERVICES 2022* . IEEE, Jul. 2022, pp. 2–10.

[12] K. Fu, W. Zhang, Q. Chen, D. Zeng, and M. Guo, "Adaptive Resource Efficient Microservice Deployment in Cloud-Edge Continuum," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1825–1840, Aug. 2022.

[13] A. Orive, A. Agirre, H.-L. Truong, I. Sarachaga, and M. Marcos, "Quality of Service Aware Orchestration for Cloud–Edge Continuum Applications," *Sensors*, vol. 22, no. 5, p. 1755, Jan. 2022.

[14] V. Casola, A. D. Benedictis, S. D. Martino, N. Mazzocca, and L. L. L. Starace, "Security-Aware Deployment Optimization of Cloud–Edge Systems in Industrial IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 724–12 733, Aug. 2021.

[15] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Sep. 2017.

[16] M. Anisetti, C. A. Ardagna, E. Damiani, F. Gaudenzi, and G. Jeon, "Cost-effective deployment of certified cloud composite services," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 203–218, 2020.

[17] M. Anisetti, C. A. Ardagna, F. Berto, and E. Damiani, "A security certification scheme for information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2397–2408, 2022.

[18] M. Anisetti, C. A. Ardagna, and N. Bena, "Multi-dimensional certification of modern distributed systems," *IEEE Transactions on Services Computing*, pp. 1–14, 2022.

[19] C. A. Ardagna, V. Bellandi, M. Bezzi, P. Ceravolo, E. Damiani, and C. Hebert, "Model-based big data analytics-as-a-service: Take big data to the next level," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 516–529, 2021.

[20] ETSI, *GS MEC 003 Multi-access Edge Computing (MEC); Framework and Reference Architecture*, V3.1.1, Mar. 2022.

[21] F. Giannone, P. A. Frangoudis, A. Ksentini, and L. Valcarenghi, "Orchestrating heterogeneous MEC-based applications for connected vehicles," *Computer Networks*, vol. 180, p. 107402, Oct. 2020.

[22] K. Liolis, J. Cahill, E. Higgins, M. Corici, E. Troudt, and P. Sutton, "Over-the-air demonstration of satellite integration with 5G core network and multi-access edge computing use case," in *IEEE 5G World Forum, 5GWF 2019 - Conference Proceedings*, Sep. 2019, pp. 1–5.

[23] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G networks," *ETSI white paper*, vol. 28, pp. 1–28, 2018.

[24] M. Anisetti, C. A. Ardagna, and E. Damiani, "Security Certification of Composite Services: A Test- Based Approach," in *2013 IEEE 20th International Conference on Web Services*, 2013, pp. 475–482.