The Cluster-based Architecture of Ferrovia 4.0

Bernardo Cabral[1,2] , Pedro Costa[1,2], Tiago Fonseca[1,2], Luis Lino Ferreira[1,2], Luis Miguel Pinho[1,2], Ricardo Severino[1,2] and Pedro Ribeiro[3]

[1] INESC-TEC, Porto, Portugal
[2] ISEP/IPP Instituto Superior de Engenharia do Porto, Portugal
{bemac, pscta, calof, llf, lmp, sev}@isep.ipp.pt;
[3] EVOLEO Technologies, Porto, Portugal
pedro.ribeiro@evoleotech.com

Abstract. This work introduces Edge4CPS, an open-source multi-architecture solution built over Kubernetes, for project Ferrovia 4.0, that aims to enable an easy to use, efficient and scalable solution for the deployment of applications on edge-like distributed computing clusters. The solution allows for a quick and easy deployment of applications on a multi-architecture cluster, already supports a few generic functionalities like a data processing pipe-line and communications middleware brokers, as well as a multi-protocol middleware translator.

Keywords: Cyber-Physical Systems · Edge Computing · Kubernets

1 Introduction

Edge4CPS [1] has been designed with the objectives of enhancing scalability, achieving efficient processing, easy handling of data, and enabling an effortless and straight-forward deployment of service applications. Out of the box, Edge4CPS delivers an API and a Deployment Interface, with pre-built components and scripts, that facilitates the interaction between the cluster and the user. Among pre-built scripts, the system delivers a straightforward Kubernetes deployment, where the user can define its nodes and their specific configurations, for exam-ple as master or worker node. Moreover, Edge4CPS also offers an effortless deployment of communication brokers and an automatic architecture-based deployment script. Regarding the pre-built service components, Edge4CPS includes a ready to use data pipeline, a database, and a middleware protocol translator. Despite being a general architecture, we will interchangeably refer to our solution by Edge4CPS or simply as edge cluster. Figure 1 shows a high-level overview of the system components.

2 The Edge4CPS Cluster

The cluster is the crucial concept behind Edge4CPS. It is implemented using Kubernetes and is responsible for hosting, coordinating, managing and scaling multiple computing nodes. Nodes within the cluster can play two distinct roles,

as a master node or as a worker node. The master node serves as the main management machine, and includes a Deployment Interface, an API, and a containerized environment.

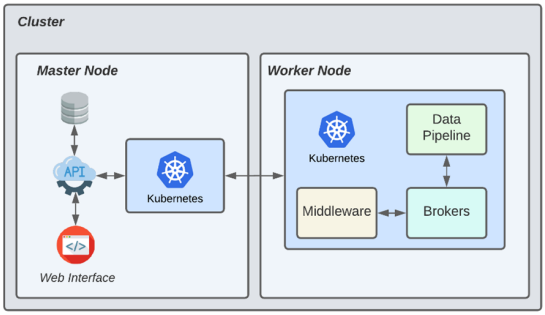

Fig. 1. Data Pipeline generic architecture components

On the other hand, the worker node typically only contains a containerized environment, increasing the capacity of the cluster to support more applications and services. The containerized environment exists on each node in the cluster, whether it be the master node or the worker node. Effectively, it is within this environment that chosen applications and essential support services run, including the environment monitoring dashboard, the data pipeline for data processing, and the middleware that enables the possibility of translating multiple communication protocols, this environment also host all the users applications and services.

The monitoring of the cluster is supported by the Kubernetes dashboard. Since it offers various monitoring capabilities, such as cluster and node monitoring, pod monitoring, resource usage monitoring, event tracking, and logging, as well as the ability to monitor deployments and Replica Sets. Overall, the Kubernetes Dashboard offers a comprehensive set of monitoring tools for the cluster, making it easier to manage and troubleshoot all the applications and services [2].

2.1 Deployment Interface and API

The Deployment Interface (Web Interface in Figure 1) is the cornerstone of the simplified user interaction with Edge4CPS. It was developed using React with the aim of abstracting the backend API and simplifying the deployment processes through a graphical interface. It includes an authentication process and allows users to fill out a form. This form contains fields for basic configuration, such as, image name, contain-er name and ports to be opened. It also allows to configure resource usage limits, like CPU and memory.

The API, uses the form filled from the Deployment Interface (generating a deployment file) to deploy the application to the cluster. This interface also enables easy access to Kubernetes dashboard for managing the containerized environment.

The API serves as the connection point between the user-oriented Deployment Inter-face and the cluster. Its responsibilities include creating a configuration file according to the deployment type and services that the user specifies. Additionally, it is tasked with providing information about the open ports of the services within the cluster, to enable the users to connect to specific services.

To deploy a service this API uses deployment file from the web interface and sends an order to the Kubernetes Framework. It is also the API that, upon receiving the user's deployment request, automatically label the deployment file according to the processor architecture on which the image was built. This labeling will later be utilized by Kubernetes to determine the appropriate node for deploying the image, based on the node CPU architecture. To do this, the API, sends a request to the image Docker Hub repository and checks what type of architecture the image supports.

2.2 Middleware

Edge4CPS middleware aims to support communication between multiple messaging communication protocols. It supports Kafka, MQTT, AMQP, and in the future OPC-UA and others. By being compatible with the Arrowhead framework, the middleware can use its main services: Orchestration, Discovery, and Authentication to select to which brokers to connect to. If Arrowhead is not available the middleware can be manually configured with the brokers addresses, through a configuration file.

The user configures the middleware based on the publishers and the consumers that need to be connected using different brokers. For example, the user could require an MQTT publisher to redirect messages to an AMQP and to a Kafka subscriber. The application will then create its own MQTT subscriber, and both an AMQP and Kafka publisher, connect the three of them internally based on additional configurations provided by the user (for example, which topic to publish to, or additional connection details), and the now running consumers will do the rest.

2.3 Database

A database requires a pod running the database image and a storage volume. The storage volume is used to provide persistent storage for the database's data. The Data-base allows, not only the storage of high volumes of input data, but also of processed data, the results of data analysis, and specific purpose data. For the most common application Edge4CPS is aiming for, it should also be capable of processing a high volume of transactions per second, which might typically reach a data acquisition frequency of more than 2 kHz per sensor [3].

2.4 Data Pipeline

The data pipeline is a pre-built framework prepared for easy and fast deployment. Its components can be distributed across multiple CPU cores and multiple nodes, resulting in an efficient utilization of computing resources and enhanced capabilities for data processing (Figure 2). During the classification phase, in which complex AI models can be used, its deployment is automatically optimized by choosing the node that is best prepared for such workload, e.g., a node with a graphics card with CUDA cores, which is more efficient for the processing of AI workloads .

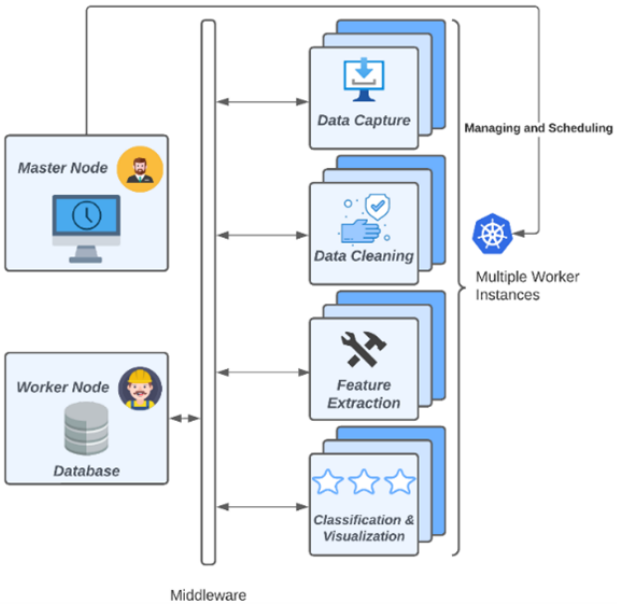

Fig. 2. Data Pipeline generic architecture components

References

1. SoftCPS. Edge4cps repository: https://github.com/bernardocabral24/edge4cps.
2. S. Muralidharan, G. Song, and H. Ko. Monitoring and managing iot applications in smart cities using kubernetes. In *CLOUD COMPUTING 2019 The Tenth International Conference on Cloud Computing, GRIDs, and Virtualization*, volume 11, 2019.
3. Pedro Chaves, Tiago Fonseca, Luis Lino Ferreira, Bernardo Cabral, Orlando Sousa, André Oliveira, and Jorge Landeck. An iot cloud and big data architecture for the maintenance of home appliances. In *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2022.