

Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing applications

Abdul Ahad Ayaz

Summary

In recent years, with the evolution of technology, IoT devices are increasing day by day. These IoT devices serve the mankind in different ways such as smart cities, smart transportation, water and waste management. With this increasing number of devices, the management and communication between these devices is one of the main issues. Fog computing addresses this issue by connecting heterogeneous IoT devices in a network and all the processing is done at the device level. Fog computing is the extension of cloud computing but decentralized.

Fog computing is responsible for providing resources to IoT devices for processing. Traditionally these resources are allocated as VMs from different cloud infrastructure such as AWS, Google, OpenStack etc. to run the applications. VMs are considered resource greedy and require more computational resources. An alternate is to use the Containers such as Docker[cite] which are light-weight, require less resources and based on micro-service architecture. Large applications are split into containers based on the main processes of the application. This increasing number of containers per application requires the proper monitoring for health check and resource consumption. The most commonly used orchestrator for containers is Kubernetes[cite].

Kubernetes is an open-source platform for management, deployment and scaling of containers. Kubernetes architecture is based on master-slave model. It consists of one master node and multiple worker nodes. Worker nodes provide the compute power. Master node communicates with the worker nodes using API, it is also responsible for scheduling and deploying the containers across the cluster of worker nodes. Applications consisting of multiple containers are deployed as a pod in Kubernetes. Each pod consists of only one IP address and all the underlying containers communicate using different ports. Different pods are isolated from each other and underlying containers cannot communicate across pods. After receiving the pod configurations from user, master node schedules these pods based on the resources availability on different worker nodes.

References