JADE Load Balancer

MATTEO MILANESE

JADE

- Java-based software framework
- Middleware that complies with the FIPA specifications
- Distributed
- Composed of containers
- Containers are the agents' runtime: the environments without which agents cannot exist

Kubernetes

- Containers and services orchestrator
- Different meaning for container:
  - Isolated user space instance
  - Application with all its dependencies
  - OS-Level virtualization (opposed to Hardware virtualization typical for VMs)
- Same concept used in Docker or containerd

Workers

- Listen for incoming job requests
- If the queue is full, signal to dispatcher that can’t accept jobs, otherwise respond positively
- Accept jobs and
  - queue them based on their priority
  - run some of them in parallel
- Note: jobs are first negotiated before being actually accepted

Dispatchers

- Retrieve worker agents by name: only agent with name starting with “worker” (with any casing) are considered
- Ask the agents if they can perform job
- If the response is positive, dispatch the job to the first available agent, otherwise repeat the steps trying to re-dispatch the same job

Containerization and Deployment

- Containerization using Docker: it creates a local image
- Local registry which stores the image
- Deployment to K8s through *Deployments*: runs multiple replicas of an application and automatically replaces any instances that fail or become unresponsive. Define *Pods* and containers in them.
- Pod communication achieved via *Services* to make JADE's containers communicate
- Services are an abstract way to expose an application running on a set of Pods as a network service

Metrics and Prometheus (1)

- Worker agents expose metrics about running and queued jobs
- K8s natively supports only few metrics
- *Prometheus* make K8s aware of agents' metrics
- It's an open-source system monitoring and alerting toolkit
- Helm (Package manager for K8s) for deployment and configuration

Metrics and Prometheus (2)

- To make K8s aware of metrics, need to deploy all Prometheus stack:
  - *Prometheus Operator:* provides Kubernetes native deployment and management of Prometheus and related monitoring components
  - *Prometheus Adapter:* provides a layer between Prometheus Operator and K8s. It contains an implementation of the Kubernetes resource metrics, custom metrics, and external metrics APIs.
- Prometheus Adapter is the key: it scrapes and exposes metrics to K8s

Autoscaling

- Horizontal Pod Autoscaler (HPA): automatically updates a workload resource with the aim of automatically scaling the workload to match demand
- Reads metrics from Prometheus, thanks to Prometheus Adapter
- Replicates Pods (scale out or in) based on defined rules
- The motivation for making Pods scaling, and in particular, Pods hosting containers running Worker agents, is the following: the basic task of a load balancer is scale based on load demand thus, if the demand increase, the instances of the application increase, and viceversa

Thank you