Assignment 3.2: Container Orchestrations

UvA - Web Services and Cloud-Based Systems

Tianhao Xu(14129027), Summer Xia(14094584), Yiming Xu(13284657)

1 Strengths

Kubernetes realizes the automatic operation and maintenance management of containerization (Docker). It uses Docker containers, so processes do not interfere with each other.

Kubernetes automates containerization's operation and maintenance management (Docker). It utilizes Docker containers to prevent processes from interfering with one another. [2]

In addition, it solves a number of Docker issues, such as the inability to effectively cluster, the lack of a centralized configuration management center tool, the absence of a container life cycle management tool, etc. A Pod is the smallest unit that Kubernetes can create or deploy.

Kubernetes has a self-healing feature, which consists of restarting failed containers, replacing and redeploying them when a node fails, and ensuring the expected number of replicas (3 containers running). It will not process client requests until it is ready, ensuring that online service is not interrupted. And it can mobilize resources in a flexible manner. For instance, when a node's CPU or memory is insufficient, the newly created pod will be assigned to the node that has been expanded. Kubernetes is generally based on Docker, which implements the operation, maintenance, and management of multiple containerized programs, thereby increasing Docker's effectiveness.

2 Drawbacks

However, k8s is not friendly to newbies, and the first deployment will take a lot of time to complete the correct configuration. [3] And as a third-party management system for containers, the services provided by K8s are affected by the defects of the container itself.

3 Our Experience

Based on the barebone(reference implementation) from assignment 2, we created a Control node and two Worker nodes on the k8s cluster.

For control node 038, we used kubeadm to setup the control plane. After the configuration, the token can be gained. We have chosen 037 and 039 as the worker nodes. Used the worker nodes to join the control node 038. For worker node 037, which is used for URL shortener. For worker node 039, which is used for User Authentication.

As for the deployment, for each app (url-shortener and user-auth), we created three pods on the master node, which is set to 3 replicas (six in total). Then, after creating a Service to include them, use the nodeport method to map the port of the Service to a port that can be accessed by the public network, so that our two microservices can be accessed through the network.

In the YAML file, we assign service port 5002 to the url-shortener function and port 5001 to the user-auth function. To expose the Service to public network, we set nodePort 30001 and 30002 on the master nodes.

In Kubernetes, a Pod represents a cluster-based process. A Pod contains multiple containers. On the same Docker host, the containers in a Pod share network, storage, and computing resources. Containers within the same Pod can communicate with each other via localhost. Ingress manages HTTP-based external access to cluster services. We use ingress for Load Balancer. Usually, Pods are built in the network covered by Ingress by default. They are associated with each other through services. The load balancing of Pods is realized through ingress Controller, and requests for services are transmitted to the target Pod through ingress.

4 Conclusion

An ingress is a resource type in Kubernetes that defines the rules for how requests are forwarded to a Service. We used a nginx-based Ingress Controller [1], which parses the Ingress-defined rules and forwards requests according to the configured rules, enables load balancing.

5 Contribution

Each team member was extensively involved in the project.
Summer Xia: Created Deployments and Service, Completed the testing.
Yiming Xu: Finished the report.
Tianhao Xu: Finished settings of docker, k8s and control & worker nodes.

References

1. "NGINX Ingress Controller". [Online] (2020), available: https://kubernetes.github.io/ingress-nginx/ [Accessed MAY.09, 2022]
2. Hardikar, S., Ahirwar, P., Rajan, S.: Containerization: Cloud computing based inspiration technology for adoption through docker and kubernetes. In: 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC). pp. 1996–2003. IEEE (2021)
3. Luksa, M.: Kubernetes in action. Simon and Schuster (2017)