Assignment 4: Exploring Kubernetes

CS328 - Distributed and Cloud Computing

DDL: 23:59, December 29, 2024

1 Introduction

In this assignment, you need to set up a local **Kubernetes (K8s)**[1] cluster using **Kind**[2]. In this K8s cluster, you will configure simple deployments and experience different pod scheduling methods. You will use `kubectl`[3] - a command line K8s client, to interact with the K8s cluster.

2 Tasks

You are given 3 tasks, where the first 2 tasks are K8s-relevant, and the final task is an optional bonus about Cloud Computing.

2.1 Task 0: K8s Deployment & Service

Start with the RESTful API Python Flask[4] server implementation from the Reverse Proxy - Load Balancing demo in a previous lab. Now, modify the root API (i.e., `/`) of the Flask server to return a hello message that includes: (1) pod name, (2) pod IP, and (3) node name (where the pod is deployed). You may find this official documentation page helpful.

Besides, the Flask server needs to support graceful shutdown to ensure that pod termination proceeds smoothly. You may find relevant information from this blog.

Manually build the image of the modified Python Flask server via `Dockerfile`. Specify a tag (e.g., `1.0.0`) for the image to represent the version of the server app.

Use Kind to create a 4-node cluster (1 control plane + 3 worker nodes). Write the cluster configurations to a YAML file. Since Kind manages nodes inside docker containers, you need to further load your local docker images into the Kind cluster nodes. You may find this official documentation page helpful.

Now, use declarative object configuration to write a YAML file that includes: (1) 1 K8s deployment that manages 4 pod replicas - each having 1 Flask server, and (2) 1 K8s

[1]Kubernetes (K8s): `https://kubernetes.io/`
[2]Kind: `https://kind.sigs.k8s.io/`
[3]`kubectl`: `https://kubernetes.io/docs/reference/kubectl/`
[4]Python Flask: `https://flask.palletsprojects.com/en/stable/`

service to expose the deployment to nodes within the cluster. Apply this configuration file, access the control plane node, and send multiple requests to the exposed service to test the modified root API. Analyze how requests are handled by the service.

Then, manually kill one server pod. Analyze what happens using `kubectl` commands.

Finally, update the Flask server implementation to add the `greet-with-info` API (i.e., `/chat/{username}`) from the RESTful API - Hello World demo in a previous lab. Build the image again but with a new version tag (e.g., `1.0.1`). Roll out the update to the existing K8s deployment by switching the image version. Show the events and operations triggered for the K8s deployment via `kubectl describe`. Explain how the rollout is handled and why by examining `maxUnavailable` and `maxSurge`. Also, verify that the server has been updated with the added API.

As a summary, you need to provide:

1. A `flask_app/` folder:
   (a) Source code of the Flask server app: `app.py`.
   (b) Build context: `requirements.txt` and `Dockerfile`.
2. A `t0/` folder for Task 0:
   (a) Kind cluster configuration: `kind-config.yaml`.
   (b) K8s resource configuration: `t0.yaml`.

Briefly describe in your report how you follow the aforementioned instructions (i.e., used commands & operations). Be sure to put all your analysis in the report and include screenshots when necessary.

2.2 Task 1: K8s Pod Scheduling

Reuse the updated Flask server (with 2 APIs) in the previous task. Use Kind to create a 6-node cluster (1 control plane + 5 worker nodes). Write the cluster configurations to a YAML file. The following labels and taints are pre-assigned to the cluster nodes:

1. Labels for worker nodes:
   (a) (Worker 1) `usage: normal`
   (b) (Worker 2) `usage: normal; capability: powerful`
   (c) (Worker 3) `usage: normal; capability: powerful`
   (d) (Worker 4) `usage: backup`
   (e) (Worker 5) `usage: backup`
2. Taints for worker nodes:
   (a) (Worker 1) NONE

(b) (Worker 2) NONE

(c) (Worker 3) if the pod does not tolerate `class=vip`, then it cannot be scheduled here

(d) (Worker 4) NONE

(e) (Worker 5) NONE

Now, use declarative object configuration to write a YAML file that includes: (1) 1 K8s deployment that manages multiple pod replicas (initially set to 1) - each having 1 Flask server, and (2) 1 K8s service to expose the deployment to nodes within the cluster. The deployment needs to satisfy the following constraints:

1. Pod Anti-Affinity: Pod replicas should not be deployed to the same cluster node.
2. Preferred Weighted Node Affinity: If there are still powerful nodes, deploy to these nodes; otherwise, deploy to nodes that are not for backup purposes; otherwise, fall back to using backup nodes.

Apply this configuration file. Start with 1 replica and gradually scale out to 5 using `kubectl scale`. Record and analyze the pod scheduling results via `kubectl describe`. In your report, state whether all pod replicas are successfully scheduled to the cluster nodes when the number of replicas is 5. If not, provide a solution by improving the configuration of the K8s deployment. Specifically:

1. The previous constraints for the K8s deployment should remain unchanged.
2. After the improvement, 5 pod replicas should be successfully scheduled onto all 5 different worker nodes.

As a summary, you need to provide:

1. A `t1/` folder for Task 1:
    (a) Kind cluster configuration: `kind-config.yaml`.
    (b) K8s resource configuration: `t1.yaml`.

Briefly describe in your report how you follow the aforementioned instructions (i.e., used commands & operations). Be sure to put all your analysis in the report and include screenshots when necessary.

2.3 Bonus (*): Advice on Future Cloud Computing Lab

This is an optional task. In your report, briefly brainstorm some ideas for the lab content on Cloud Computing. What relevant topics are you interested in learning? What kinds of lab exercises would you like to try? Based on the coherence, prevalence, and feasibility of your answers, you will get 5 bonus points at max. **If you get more than 100 points in total, your bonus points will be scaled and added to the overall score of your homework.**

3 Submission

Be sure to include in your submission: source code files and a report (using a provided template) in **PDF** format. Pack all files into `SID_NAME_A4.zip`, where `SID` is your student ID and `NAME` is your pinyin name (e.g., `11710106_ZhangSan_A4.zip`). Any text should be in English only. Plagiarism is strictly prohibited.