Lab 12: Kubernetes ConfigMaps

Overview

In this lab, you'll delve into Kubernetes ConfigMaps, focusing on managing non-confidential data and upgrading your application for persistence. ConfigMaps provide a way to decouple configuration artifacts from image content, allowing you to manage configuration data separately from the application.

Task 1: Upgrade Application for Persistence

6 Points:

In this task, you'll enhance your application to persist data and explore ConfigMaps in Kubernetes.

1. Upgrade Your Application:
    - Modify your application to:
        - Implement a counter logic in your application to keep track of the number of times it's accessed.
        - Save the counter number in the `visits` file.
        - Introduce a new endpoint `/visits` to display the recorded visits.
    - Test the changes:
        - Update your `docker-compose.yml` to include a new volume with your `visits` file.
        - Verify that the enhancements work as expected, you must see the updated number in the `visits` file on the host machine.
        - Update the `README.md` for your application.
2. Create Pull Requests:
    - Submit a PR to merge your changes into the main branch of the forked repository.
    - Create a PR from the `lab12` branch to the main branch in your own repository.

Task 2: ConfigMap Implementation

4 Points:

1. Understand ConfigMaps:
    - Read about ConfigMaps in Kubernetes:
        - ConfigMaps
2. Mount a Config File:
    - Create a `files` folder with a `config.json` file.
    - Populate `config.json` with data in JSON format.
    - Use Helm to mount `config.json`:
        - Create a `configMap` manifest, extracting data from `config.json` using `.Files.Get`.
        - Update `deployment.yaml` with `Volumes` and `VolumeMounts`.

        - Example
    - Install the updated Helm chart and verify success:
        - Retrieve the list of pods: `kubectl get po`.
        - Use the pod name as proof of successful deployment.
        - Check the ConfigMap inside the pod, e.g., `kubectl exec demo-758cc4d7c4-cxnrn -- cat /config.json`.

3. Documentation:

    - Create `12.md` in the `k8s` folder and include the output of relevant commands.

List of Requirements:

- `config.json` in the `files` folder.
- `configMap` retrieving data from `config.json` using `.Files.Get`.
- `Volume`s and `VolumeMount`s in `deployments.yml`.
- `12.md` documenting the results of commands.

Bonus Task: ConfigMap via Environment Variables

2.5 Points:

1. Upgrade Bonus App:

    - Implement persistence logic in your bonus app.

2. ConfigMap via Environment Variables:

    - Utilize ConfigMap via environment variables in a running container using the `envFrom` property.
    - Provide proof with the output of the `env` command inside your container.

Guidelines

- Maintain clear and organized documentation.
- Use appropriate naming conventions for files and folders.
- For your repository PR, ensure it's from the `lab12` branch to the main branch.

> Note: Clear documentation is crucial to demonstrate successful data persistence and ConfigMap utilization in Kubernetes. Explore the bonus tasks to further enhance your skills.