API Gateway – Interview Assignment

Background

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

Kubernetes runs your workload by placing containers into Pods to run on Nodes. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods. Typically, you have several nodes in a cluster; in a learning or resource-limited environment, you might have just one.

Ingress exposes HTTP and HTTPS routes from outside the [Kubernetes] cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. … You must have an Ingress controller to satisfy an Ingress. Only creating an Ingress resource has no effect.

A controller tracks at least one Kubernetes resource type. These objects have a spec field that represents the desired state. The controller(s) for that resource are responsible for making the current state come closer to that desired state.

Background Materials for self-learning:

- Docker: https://www.docker.com/get-started
- Kuberenetes: https://kubernetes.io/docs/tutorials/kubernetes-basics/
  - Ingress: https://kubernetes.io/docs/concepts/services-networking/ingress/
  - Controllers/Operators: https://kubernetes.io/docs/concepts/architecture/controller/
  - Clusters: https://kubernetes.io/docs/concepts/architecture/nodes/
  - Networking: https://kubernetes.io/docs/concepts/cluster-administration/networking/
  - Container Specs: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/#container-v1-core (read container port spec carefully)
- Helm: https://helm.sh/docs/intro/quickstart/
- Kind: https://kind.sigs.k8s.io/docs/user/quick-start/
  - Ingress in kind: https://kind.sigs.k8s.io/docs/user/ingress/
- Kubebuilder book: https://book.kubebuilder.io/
- Useful Golang http libs:
  - https://golang.org/pkg/net/http/#Server
  - https://golang.org/pkg/net/http/httputil/#ReverseProxy

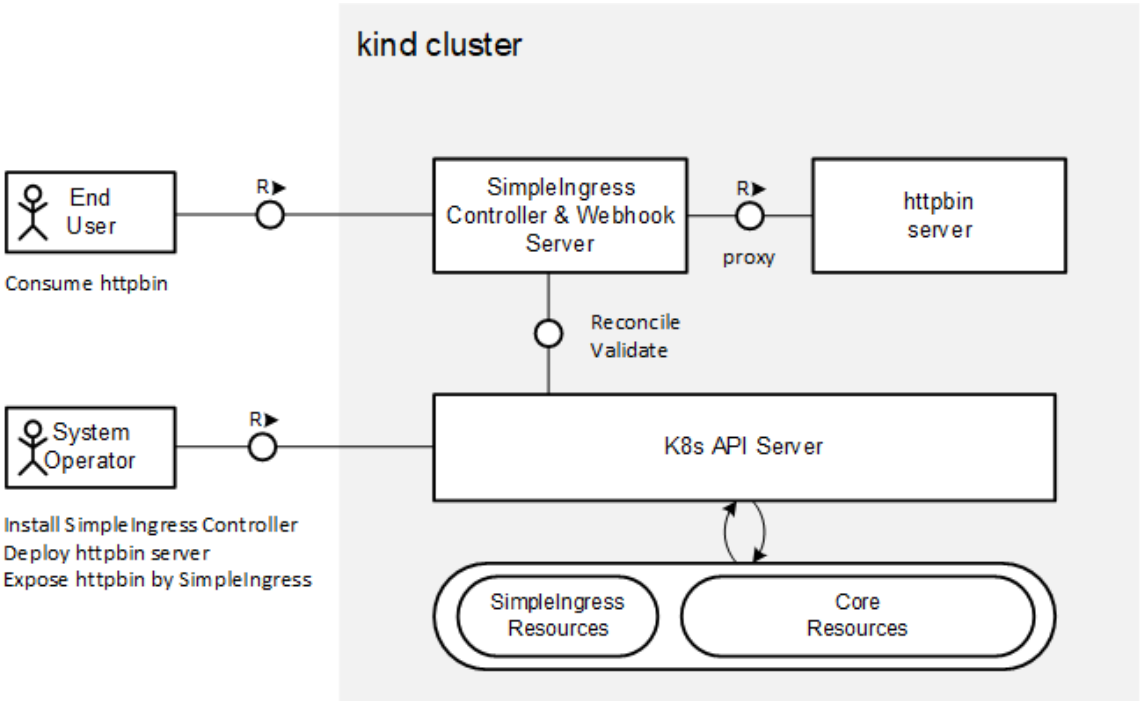
Assignment

SimpleIngress is a new K8s resource that performs a simple mapping of an external hostname to a cluster-internal k8s service name for http traffic.

- Each resource contains a single mapping (rule).
- *SimpleIngress* is a namespaced resource, that can only refer to k8s services in the same namespace.
- By applying *SimpleIngress* resources, your consumers can expose their cluster-internal services externally.

Your mission it to build a *SimpleIngress* controller running on a local kind (Kubernetes-in-Docker) cluster and use it to expose a cluster-internal http server with a *SimpleIngress* resource.

High-Level Architecture of the assignment



Implementation guidelines

- Use the *KubeBuilder* framework to create your *SimpleIngress* Controller
- Design the *SimpleIngress* Custom resource schema to fit the requirements.
- Your controller should reconcile the above-mentioned resource and configure an http reverse proxy to route http traffic according to the rules.
- Apply the controller and webhook patterns to handle additional considerations:
  - Multiple rules may be added, updated and deleted
  - Your consumers should have a way to know if the rule is applied successfully
  - Validate SimpleIngress resources

Clarifications

- Your controller can behave as the reverse http proxy itself
- Your solution should be containerized with docker and deployable with helm
- Only basic routing abilities are required:
    - Only plain HTTP/1.1 support is required (no need for HTTP/2, HTTPS, WS, WSS, etc.).
    - Only port 80 is in use and it is not explicitly stated in any address (external or internal).
    - Only basic host-based routing is required, no need to implement advanced features like path, host and regex-based routing.
- The solution should work locally on a kind cluster
    - Your Ingress controller is the only component that is exposed externally to the cluster.
    - Understand how other kind ingress controllers are exposed, useful materials here: https://kind.sigs.k8s.io/docs/user/ingress/ , https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/#container-v1-core
    - You can use IP DNS wildcard services like https://nip.io/ to create addressable hostnames, eg. your ingress rules can look like this: *http://httpbin.<my-ip>.nip.io -> httpbin-svc*.

This assignment was validated using the following dependency versions, which should also work for you:

- docker 19.03.12
- golang 1.14
- kind v0.10.0
- kubebuilder v2.3.1
- kustomize v3.6.1

Results

- Provide a zip file with all sources of your solution (code, deployment descriptors, etc.)
- Deploy your docker image to: https://hub.docker.com/, and chart to: https://artifacthub.io/
- Provide a shell script that runs your solution end-to-end and demonstrates functionality:
    - create a kind cluster
    - deploy your controller and any additional dependencies that you have
    - deploy a simple echo server (eg. https://github.com/hashicorp/http-echo) and expose it with a service and a *SimpleIngress*
    - curl the echo server after the ingress resource has been reconciled