KUBERNETES ORCHESTRATION

INGRESS

INGRESS – OVERVIEW

- Provides external access to services in your cluster. Usually HTTP
- Can provide load balancing, SSL, and name based virtual hosting.
- Can specify complex routing rules
- You can choose many different controllers to provide the functionality, including:
  - Nginx (maintained by K8S project)
  - Contour/Envoy
- Popular use is to centralize many microservices under one name using routing rules (AKA edge service or API gateway). For example:
  - Example.com/account points to account service
  - Example.com/orders points to order service
- Other solutions exist for this. For example, cloud providers have their own L7 load balancers such as Application Load Balancer from AWS. Ingress is just the K8S solution. You can also do it internally with your own app like we will do in the hackathon portion.

INGRESS – OVERVIEW

- You need two main components:
  - Ingress controller – provides the actual processing logic. Kubernetes does not ship with a controller. You must implement your own. Many options exist for this, including Nginx, which we'll use.
  - Ingress – Your own set of rules that tell the ingress controller how to behave. This is a resource in your own application.

INGRESS – LAB

Let's start with an overview of the RV Store application. We're going to replace the custom developed API Gateway application with an Ingress.

1. Review ingress/ingress-controller.yaml. Note the Namespace, ConfigMaps, RBAC resources, and workloads. Apply it.
2. Review ingress/ingress-service.yaml. This is the service for the Deployment in the controller. Apply it.
3. Review ingress/ingress.yaml. This is our actual Ingress, which sets up the path routing rules. Apply it.
4. Apply all files in the minikube directory. This will deploy the application to your workstation, including the built-in gateway.
5. Open http://localhost:30080 in your browser. In the "Backend host name" field, enter http://localhost:30090/ and click Update. You should see five products. This is using the built-in gateway.
6. Refresh the page. Now enter http://localhost:30880/ into the Backend host field and click Update. You should see the products again. But this time, your traffic was routed through the Ingress.