README

Kubernetes Gateway API Demo Goal Learn the core Kubernetes Gateway API objects with a runnable project: - `GatewayClass` (provided by your Gateway controller) - `Gateway` (data-plane entry point) - `HTTPRoute` (traffic routing rules) This demo uses path-based routing: - `/v1` -> `echo-v1-svc` - `/v2` -> `echo-v2-svc` - `/` -> `echo-v1-svc` Project structure `text gateway-demo/ README.md manifests/ 00-namespace.yaml 01-apps-services.yaml 02-gateway.yaml 03-httproute.yaml` Kubernetes Gateway concept (quick notes) **Gateway API replaces many Ingress limitations** It is role-oriented and extensible. `GatewayClass` Cluster-scoped definition of which implementation handles Gateways (for example: NGINX Gateway Fabric, Istio, Kong, Traefik). `Gateway` Namespaced listener configuration (ports/protocols/TLS) that attaches to a `GatewayClass`. `HTTPRoute` Route rules (host/path/header matching, filters, backends) attached to one or more Gateways. **Separation of concerns** Platform team can manage infra (`GatewayClass`, shared `Gateway`). App teams can manage routes (`HTTPRoute`) in their namespaces. Prerequisites Running Kubernetes cluster `kubectl` configured Gateway API CRDs installed A Gateway controller installed and running Gateway setup instructions for existing Minikube (including CRDs + controller install): see `gateway-demo/installation.md`. Check support: `bash kubectl api-resources | findstr gateway.networking.k8s.io kubectl get gatewayclass` If no `GatewayClass` exists, install your preferred Gateway controller first. Important before apply In `manifests/02-gateway.yaml`, this demo uses: `yaml gatewayClassName: nginx` If your cluster uses a different class, replace `nginx` with the output from: `bash kubectl get gatewayclass` Deploy `bash kubectl apply -f gateway-demo/manifests/00-namespace.yaml kubectl apply -f gateway-demo/manifests/01-apps-services.yaml kubectl apply -f gateway-demo/manifests/02-gateway.yaml kubectl apply -f gateway-demo/manifests/03-httproute.yaml` Or all at once: `bash kubectl apply -f gateway-demo/manifests/` Verify `bash kubectl get deploy,svc -n gateway-demo kubectl get gateway,httproute -n gateway-demo kubectl describe gateway gateway-demo -n gateway-demo kubectl describe httproute echo-route -n gateway-demo` Look for: - `Gateway` accepted/programmed by controller - `HTTPRoute` accepted and attached to `gateway-demo` Test traffic Find the Gateway data-plane address from your controller service (often type `LoadBalancer`): `bash kubectl get svc -A` Then test: `bash curl http://<GATEWAY-ADDRESS>/v1 curl http://<GATEWAY-ADDRESS>/v2 curl http://<GATEWAY-ADDRESS>/` Expected responses: - `/v1` -> `echo-v1` - `/v2` -> `echo-v2` - `/` -> `echo-v1` Useful debug commands `bash kubectl get gatewayclass kubectl get gateway -n gateway-demo -o yaml kubectl get httproute -n gateway-demo -o yaml kubectl get events -n gateway-demo --sort-by=.lastTimestamp kubectl logs -n gateway-demo deploy/echo-v1 --tail=20 kubectl logs -n gateway-demo deploy/echo-v2 --tail=20` Cleanup `bash kubectl delete -f gateway-demo/manifests/03-httproute.yaml --ignore-not-found=true kubectl delete -f gateway-demo/manifests/02-gateway.yaml --ignore-not-found=true kubectl delete -f gateway-demo/manifests/01-apps-services.yaml --ignore-not-found=true kubectl delete -f gateway-demo/manifests/00-namespace.yaml --ignore-not-found=true`